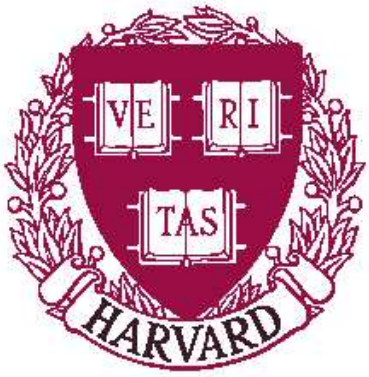
Reference Specification in Multilingual Document Production

Reference Specification in Multilingual Document Production

Jill Suzanne Nickerson

TR-21-05



Computer Science Group
Harvard University
Cambridge, Massachusetts

Reference Specification in Multilingual Document Production

A thesis presented

by

Jill Suzanne Nickerson

to

The Division of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

May 2005

To my parents, Marianne and William Nickerson. ♡

Thesis advisor

Barbara J. Grosz

Author

Jill Suzanne Nickerson

Reference Specification in
Multilingual Document Production

Abstract

To produce documents in multiple languages automatically requires a language-independent representation of the documents' meaning. For a person to build this language-independent representation by communicating in natural language with a computer system, the problem of reference must be addressed. This problem, inherent in natural language, presents itself not only in the specification of the language-independent representation, but also in the generation of documents with the meaning contained in this representation.

This thesis presents methods to make both the specification of entities in the user interface and the generation of expressions to refer to these entities in documents more natural and provides empirical evidence demonstrating the efficacy of these methods. More specifically, this thesis describes the development of three types of reference mechanisms: a statistical model that uses domain and lexical knowledge to organize new options in the interface; techniques for controlling coreference specification that take advantage of discourse structure and genre features; and automatically learned models for generating expressions to refer to new and already mentioned entities in a particular domain. The evaluation of these reference mechanisms establishes

that specifying new entities using an interface informed by computational linguistic processing reduces the amount of time required to refer to entities in the interface; exploiting discourse structure and genre features is more helpful than traditional knowledge-editing interfaces for referring to entities in the interface that are already contained in the knowledge representation; and using learned linguistic information to generate referring expressions in documents leads to expressions that more closely match the decisions of people.

Contents

List of Figures

List of Tables

Acknowledgments

First and most importantly, I am indebted to my thesis advisor Barbara Grosz for inspiring and challenging me. Barbara helped me to clarify my thinking and was always able to offer me direction and constructive comments. Barbara, you will continue to serve as a lifelong role model.

I am also grateful to the other members of my thesis committee for their time and guidance. Stuart Shieber is an expert at fleshing out ideas and looking at things in new ways. I would not have been able to develop the ideas in this thesis without his input. I would like to acknowledge Avi Pfeffer for his research suggestions as well as the opportunity to hear his musical compositions.

Maxwell-Dworkin Laboratory, Room 217 served as a relaxing place to do work thanks to my wonderful officemates. Tim Rauenbusch is a great friend and listener who is always there when I need him. Our chips and French onion dip breaks were most enjoyable. Tim, thank you for sharing your gourmet recipe. Ya'akov (Kobi) Gal has a contagious work ethic that led me to push myself harder. Kobi, I will always remember our visit to the house of Mother and Father Divine and the theatrical performance we attended in Philadelphia. Emir Kapanci's pleasant and upbeat personality helped get me through long days. Emir, thanks for the statistics lessons. Drive carefully.

Thanks to many other people affiliated with the AI research group who made Harvard such a comfortable place throughout my tenure as a graduate student. Anna Olevsky, I am glad we were able to experience the ups and downs of the first few years of graduate school together. Without your friendship, I would not have survived. It was a pleasure serving as a teaching fellow with Sanmay Das. Sanmay, thanks

for answering my questions and clearly explaining material that was unfamiliar to me. You will make a great professor, but your martinis need some work. Rebecca Hwa provided encouragement and was always happy to share lessons she had learned firsthand. She also taught me how to make dumplings. I admire Wheeler Ruml's structure and organization in his daily routine. Wheeler, congratulations on the birth of your daughter Julia. Marco Carbone always took the time to inquire about my research and well-being even when he was busy with teaching and his own projects. Best of luck in law school, Marco. Peter Arvidson was instrumental in organizing meetings with Barbara. I appreciate Peter's promptness and his friendliness.

Several people influenced me and served as mentors in my early education. Jennifer Mudrock and Al Logsdon introduced me to Pascal, my first programming language, when I was a student at Lincoln-Way High School in New Lenox, Illinois. writeln('Thank you.');

At the University of Pennsylvania, Bonnie Webber advised me on my first research project. She introduced me to the science of Artificial Intelligence and taught me the basics of conducting research. Max Mintz encouraged me to pursue graduate school and instilled confidence in me when he told me I would make the "perfect graduate student."

My internship at Lucent Technologies in the Dialogue Systems Research Department was invaluable for preparing me for graduate school. Working with top researchers at Lucent was stimulating and fun. I am grateful to my advisor Jennifer Chu-Carroll for teaching me how to work with and learn from others and how to write technical papers. Thanks also to others whom I worked with at Lucent, particularly

Bob Carpenter, Jim Hieronymus, and Jennifer Venditti.

I acknowledge the people at the ITRI for providing WYSIWYM. Donia Scott made me feel most welcome when I visited Brighton. Thanks to Richard Power for helpful discussions and suggestions. I am grateful to Gabriela Cavaglia for introducing me to the UK's most famous dish–fish and chips. Thanks also to Paul Piwek and Kees van Deemter.

I am grateful to the people at General Motors for providing me with access to their corpus of car repair procedures. I am grateful to to Ramasamy (Samy) Uthurusamy for inviting me to visit General Motors and for giving me insight on how a system for producing natural-language documentation could be used in the real world. Thanks also to Kurt Godden.

I would like to thank several friends and family members for their support. I am grateful to Marko Lončar for Pierce Coffee Sessions (PCSs). Marko, taking breaks with you brightened my afternoons. I benefited from sharing graduate school experiences with Ken Stanley. Ken, if you hadn't been there to help me with the graduate school application process, I most certainly would have been lost. Thanks to Jill D. Nickerson for inviting me to visit her in Tulsa and Los Angeles. These getaways helped to take my mind off of the pressures of graduate school. Jill, thanks for all of the advice along the way; you're the big sister I never had. I would like to acknowledge Gavin Haentjens for providing me with outside opinions on important life matters. Gavin, your observations and recommendations help me make more effective decisions. I am grateful to Grandma Nickerson and Ruth Audrey Balaban for being great pen pals throughout the years. In a world dominated by the immediacy

of email, I look forward to receiving your letters in my mailbox.

Finally, I want to offer special thanks to my parents for their encouragement, patience, and love. I appreciate and love you more than you will ever know. I dedicate this thesis to you both.

This work described in this thesis has been supported by the National Science Foundation under Grant Nos. IIS-9978343, IIS-9811129, and IIS-9618848; an NSF Graduate Student Fellowship; and a Lucent Technologies GRPW Grant.

Chapter 1

Introduction

Today's global economy requires knowledge to be shared by people around the world, people who speak many languages. To ensure that products can be used safely and effectively by people in all markets in which they are sold, manufacturers must produce documentation in many languages. This requirement has led to a demand for high-quality multilingual documentation. To produce documents in multiple languages automatically[1] requires a language-independent representation of their meaning. Two common approaches for arriving at this type of representation are for a person knowledgeable about the domain and task to specify it directly or for a system to extract it automatically from a natural-language document. Recent research has focused on an approach that lies in the middle of these two extremes, retaining advantages of each of them. In this approach, a person interacts with a computer system through an interface to construct the language-independent representation. If, as part of the dialogue in communicating the content of this representation, text

[1]There are also manual approaches for producing documents in multiple languages. These approaches are labor-intensive and, thus, very costly.

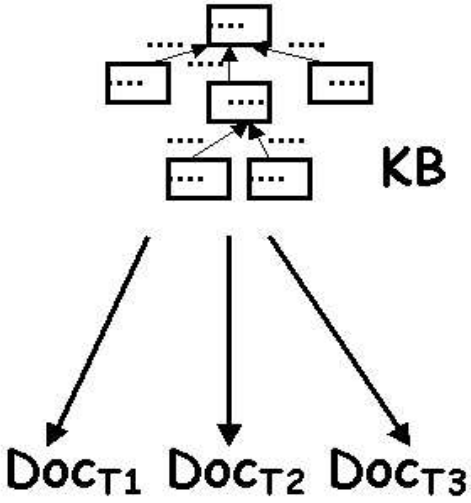


Figure 1.1: Automatically Producing Documents in Multiple Languages: Traditional MNLG Approaches.

is used, several challenges inherent in natural language arise. This thesis addresses one such challenge, the problem of reference.

1.1 Overview of Approaches for Automatically Producing Documents in Multiple Languages

Multilingual natural-language generation (MNLG), shown in Figure 1.1, is a knowledge base (KB) specification approach in which knowledge engineers specify a KB directly. Knowledge engineers not only must be domain experts, or work closely with one, but also must be proficient in the knowledge representation language, the unambiguous logical formalism used for encoding knowledge, being used to specify the KB. The KB is used to produce target documents in different languages (Doc_{T1}, Doc_{T2}, and Doc_{T3}). Because the content of a document is specified directly

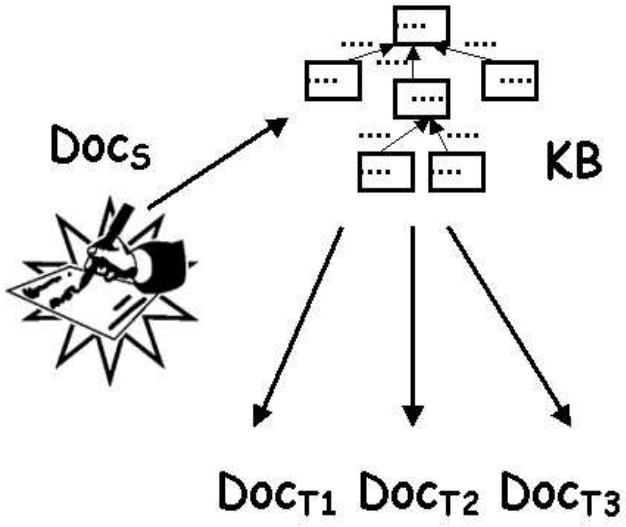


Figure 1.2: Automatically Producing Documents in Multiple Languages: Interlingual Model of MT.

and unambiguously by knowledge engineers, documents produced using MNLG approaches avoid many of the disadvantages associated with approaches in which a system extracts meaning automatically from a natural-language document (Scott & Evans 1998).

One such KB specification approach that extracts document content automatically is Machine Translation (MT), a document-oriented approach for producing multilingual documentation. This document orientation is illustrated by the interlingual model of MT, a common MT approach, which is portrayed in Figure 1.2. A technical writer composes, in a natural language, a source document, Doc$_S$. The system analyzes the meaning of the source document, and then, it automatically creates an interlingua,[2] an intermediary, language-independent KB. This KB is used to pro-

[2] Another common MT approach is the transfer-based approach. If the task is to translate between all pairs of a given set of n languages, however, transfer-based approaches are significantly more complex. They require translation in each direction for every pair of languages, which is $O(n^2)$. The

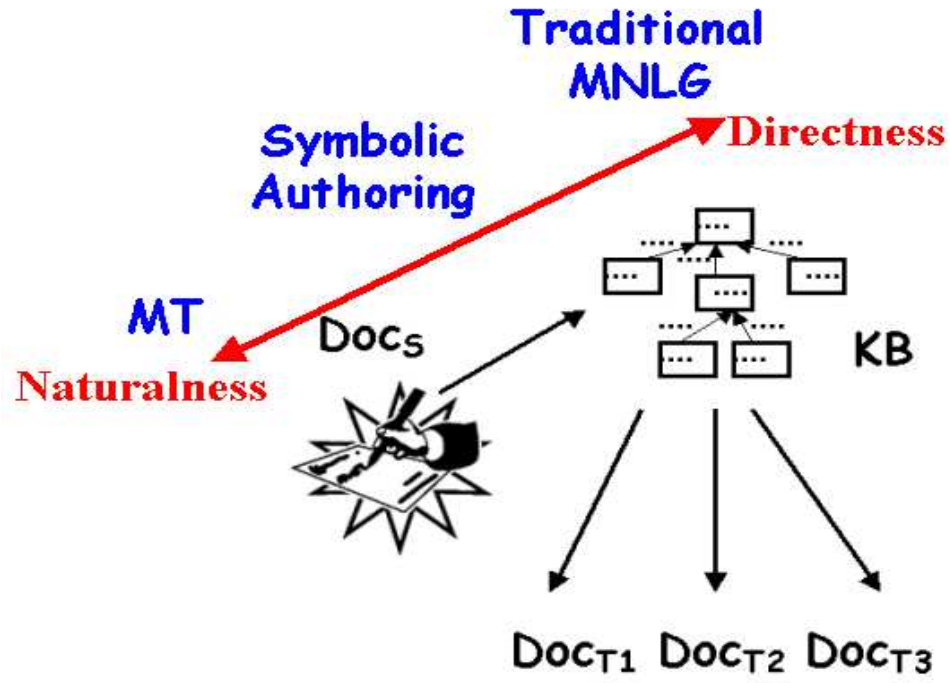


Figure 1.3: KB Specification Method Continuum.

duce target documents in different languages (Doc_{T1}, Doc_{T2}, and Doc_{T3}). The main challenge in the interlingual model of MT is extracting a KB from the source document. Constructing a useful KB is hard to perform automatically because it requires a full and accurate semantic interpretation of the natural-language source document. Often, documents produced using MT are inconsistent, over-literal, and exhibit a source-language bias. For this reason, people must often post-edit them (Scott & Evans 1998).

MNLG and MT are two end points on a continuum of KB specification methods, as illustrated in Figure 1.3. At one end is the most direct method from the point of

interlingual model of MT, on the other hand, only requires translation to and from the interlingua for each language, which is $O(n)$.

view of the system: encoding meaning in a knowledge representation language, the method used in traditional MNLG approaches. Diametrically opposed to this method is the most natural method from the point of view of a person: unconstrained natural language, the method used in the interlingual model of MT.

The directness of MNLG approaches is achieved at the expense of naturalness for knowledge engineers. Instead of writing a document in unconstrained natural language, knowledge engineers must specify the meaning of the document explicitly using a knowledge representation language, which is more constrained and more difficult to use because it is artificial. If knowledge engineers specify the KB correctly, it will be free of errors. However, specifying the KB correctly in a knowledge representation language is not trivial. In MT approaches, unlike MNLG approaches, document specifiers are unlikely to make errors, since they are specifying the document in a natural language. However, the automatic analysis the system performs on natural-language documents is prone to creating errors in the KB.

Ideally, the KB specification method should be as close to a natural language as possible, since this is the most natural specification method for people. Due to the shortcomings of the interlingual model of MT, though, it is not feasible to specify documents in unconstrained natural language. Recent research has focused on approaches that lie in the middle of the KB method specification continuum. These approaches, referred to as symbolic authoring approaches (Paris *et al.* 1995), employ a user interface in which a person and a computer collaborate to specify a KB. They retain some of the naturalness of MT approaches, but they are more direct. Early symbolic authoring approaches used a human-computer interface that was graphical

in nature (Skuce & Lethbridge 1995; Caldwell & Korelsky 1994, inter alia). WYSIWYM (Power, Scott, & Evans 1998) is an example of a symbolic authoring approach that uses a more natural representation of the KB–natural-language text. WYSIWYM presents domain experts with a user interface containing a structured dialogue consisting of a menu-based document that is easy to understand and edit. This document, which is generated by a natural-language generation system using a partial KB, communicates the current state of the KB and ways domain experts can extend it. Because experts are directly editing knowledge in a KB, the system does not need to interpret text, and the experts' selections are guaranteed to be understood by the system.

1.2 The Problem of Reference

In MNLG approaches, knowledge engineers explicitly specify to which entities they are referring as they construct a KB. Approaches for producing documents in multiple languages that use natural language in any way in the specification of the KB, however, must handle the problem of reference, a problem that arises from the inherent ambiguity of natural language. The same expression may be used in different contexts to refer to different entities. Also, reduced noun phrases and pronouns often require the surrounding context and world knowledge to be interpreted correctly. For example, the pronoun *it* is ambiguous in the following statement:

> The publication's cost will be estimated and appropriated from the budget, and when it is completed, its actual cost will be paid.

Even though *budget* is the closest noun, the intended referent of *it* is *publication*. To build an accurate KB to produce target documents that preserve the meaning of the source document, MT approaches must correctly resolve referring expressions, natural-language expressions used to refer to entities in the domain.

Like MT approaches, symbolic authoring approaches such as WYSIWYM that use natural-language text to communicate the content of the KB also have reference problems. Reference specification in WYSIWYM is more tractable than in MT approaches, since it requires references to be specified in a more direct manner. Reference specification in current systems that use WYSIWYM, however, is not very natural.

In WYSIWYM-based systems, the problem of reference arises in two places: KB specification and document generation. During the KB specification phase, the system must construct a KB containing representations that accurately reflect the entities domain experts refer to in the user interface. Figure 1.4 illustrates the problem of reference in KB specification. In this particular procedure specification, the system has incorrectly inferred that the domain expert meant to introduce a second document in Step 3 of the procedure for printing a document. This inference leads to a KB that is inconsistent with the knowledge the domain expert intended to encode; the expert intended that the document that was opened to print (and, thus, is already present in the KB) be closed.

In the generation phase, the system must produce documents containing referring expressions clearly denoting the entities to which they refer. Figure 1.5 shows an example of the problem of reference in document generation. Internally, the system

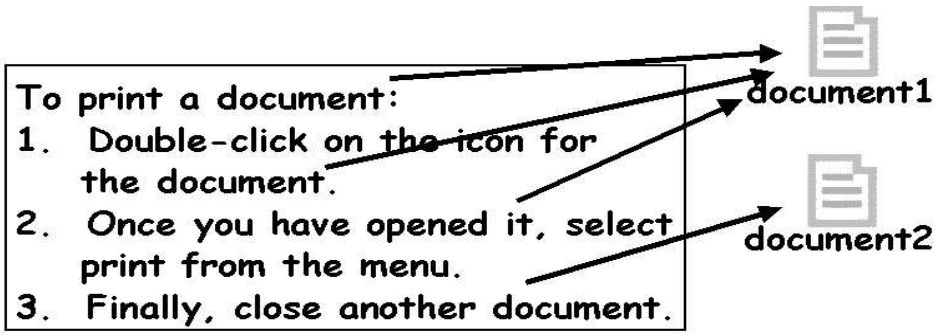


Figure 1.4: The Problem of Reference: KB Specification.

Removal Procedure:
1. Loosen the stabilizer shaft bracket bolts.
2. Remove the stabilizer shaft bracket bolts.
3. Remove the stabilizer shaft bracket.
4. Lift upwards on the stabilizer shaft insulator.
5. Remove the stabilizer shaft insulator.

Figure 1.5: The Problem of Reference: Document Generation.

has represented reference correctly. The text generated to represent the content of the KB, however, is cumbersome and, as a result, difficult to understand. In this removal procedure for the stabilizer shaft insulator, a car part, full noun phrases have been repeated in contexts where reduced noun phrases or pronouns would lead to more coherent text (Gordon, Grosz, & Gilliom 1993). For example, in Step 2, instead of repeating the full noun phrase *stabilizer shaft bracket bolts* from Step 1, the reduced noun phrase *bracket bolts* or simply *bolts* would lead to more natural-sounding text.

This thesis focuses on the problem of reference in WYSIWYM. We present methods that make the task of referring to entities more natural without losing directness,

thus moving WYSIWYM closer to the *naturalness* end of the continuum presented in Figure 1.3. The first part of the thesis examines how to make the task of referring to entities when creating a KB match the natural abilities of people. The second part provides learned rules the system can use to generate expressions to refer to entities contained in a KB to effectively match the decisions of people. To meet these requirements, the KB specification methods allow for quick and accurate reference to entities in the user interface, and the document describing the content of the KB is coherent and easily comprehensible by people.

1.3 AUTO: A WYSIWYM-based System with a Focus on Reference

To implement and test our methods for resolving the mismatch between the abilities of people and current methods used for referring to entities in WYSIWYM, we used WYSIWYM to develop AUTO, a system for producing car repair manuals. This thesis presents experiments showing that our methods lead to efficiency in KB specification and coherent documents, the content of which accurately reflects the specified knowledge.

This thesis discusses several improvements to entity reference based on computational methods for the KB specification phase of multilingual documentation production. For referring to new entities in AUTO's user interface, we use categorical organization and a statistical model based on domain and lexical knowledge. This model, which uses cooccurrence statistics collected from a corpus of car repair man-

uals, predicts those entities most often associated with verbs chosen in car repair procedure instructions. In determining the salience of entities that have already been introduced into the KB, our models take advantage of the discourse structure of the currently specified procedure and genre features of the task-specific car repair domain. This thesis also describes a number of parameters that can be adjusted to create mechanisms for specifying already mentioned entities in the user interface. These new methods are supported with user studies showing that they outperform traditional methods in terms of selection time and other important factors.

For the generation phase of multilingual documentation production, this thesis provides learned models for the generation of expressions that refer to the entities contained in the KB. The induced models consist of a sequence of rules for referring-expression generation based on the decisions of people who took part in a data collection exercise in which they were given car repair procedures and asked to choose the referring expressions that made the procedure sound most coherent. This thesis provides an evaluation of the predictions of the learned models by comparing their predications to the decisions of people who took part in the study. It shows that the predictions of the learned models lead to cohesive and fluent target documents containing natural-sounding descriptions.

1.4 Summary of Contributions

In summary, this thesis presents methods and provides empirical evidence showing how to make two aspects of reference more natural in the production of natural-language documentation: 1) the specification of entities in document preparation and

2) the generation of expressions to refer to these entities in the document produced. The application domain for implementing and empirically evaluating the mechanisms described in this thesis is the car repair domain. The algorithms and methods are employed in a system we built for authoring car repair manuals called AUTO.

We developed the following types of mechanisms:

- A statistical model that uses domain and lexical knowledge to organize new options in interfaces for specifying KBs;
- Techniques for controlling linguistic coreference in interfaces to knowledge-editing systems that take advantage of discourse structure and genre features; and
- Learned models for generating expressions to refer to new and already mentioned entities.

We performed evaluations using these mechanisms, empirically demonstrating the following:

- Specifying new entities using an interface informed by categorical organization ($p < 0.04$) and domain and lexical knowledge ($p \approx 0$) significantly reduces the amount of time required for people to refer to new entities when compared to methods used in existing knowledge-editing interfaces. Restrictive categories and informative verbs are most effective in reducing selection times.
- Appealing to discourse structure is more useful ($p < 0.04$) than traditional methods for predicting entities likely to participate in coreference relations in

instructions steps at the same level. Genre features are effective ($p \approx 0$) for referring to already mentioned entities in car repair installation procedures.

- Using learned linguistic information to generate referring expressions leads to referring expressions that more closely match the decisions of people in length and content compared to using a one-to-one mapping between the concept chosen in the interface and the referring expression used in the generated document. The learned models for referring expression generation accurately predict contexts where noun-phrase reduction is possible, thus eliminating redundancy.

1.5 Thesis Road Map

Chapter 2 provides an overview of WYSIWYM (Power, Scott, & Evans 1998), a symbolic authoring approach used to develop systems for generating multilingual documentation, and describes the development of one such functional prototype, AUTO, for generating car repair manuals. The next two chapters present methods that allow people to effectively refer to entities in interfaces to knowledge-editing systems. Chapter 3 provides methods that use categorical organization and domain and lexical knowledge to assist people in referring to new entities. The empirical study in this chapter compares these approaches to more traditional ones. Chapter 4 presents mechanisms informed by discourse structure and genre features that allow people to refer to entities that already have a representation in the KB. The effectiveness of these coreference mechanisms is also evaluated. The chapter that follows, Chapter 5, examines how the system generates expressions to refer to entities in the final

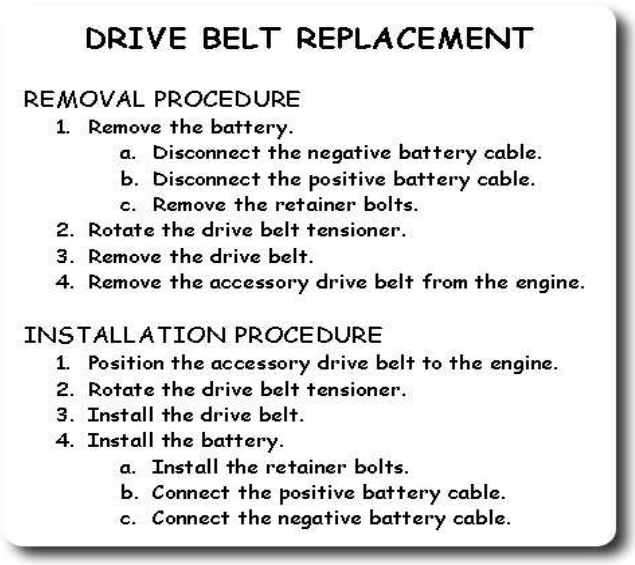
document. It focuses on the use of a transformation-based machine learning algorithm to induce models for generating referring expressions. In addition, it reports an empirical study that formally evaluates these models. Chapter 6 summarizes the main contributions of the thesis and offers concluding remarks as well as directions for future research.

Chapter 2

WYSIWYM: Overview and Application

This chapter presents an overview of the WYSIWYM (What You Say Is What You Meant) (Power, Scott, & Evans 1998) symbolic authoring approach to multilingual documentation production and describes the design and implementation of a WYSIWYM-based system called AUTO. In prior work, WYSIWYM has been applied in a number of projects: PILLS (Bouayad-Agha *et al.* 2002) to produce pharmaceutical documentation, CLIME (Piwek *et al.* 2000) to formulate queries about shipping regulations, and DRAFTER (Power, Scott, & Evans 1998) to generate software manuals. AUTO builds on the implementation of the WYSIWYM symbolic authoring approach used in DRAFTER-2 (Power, Scott, & Evans 1998), a successor of the system developed as part of the DRAFTER project.[1] AUTO is a system domain experts can use to *auto*matically generate car repair manuals. Section 2.1 walks through the

[1] The DRAFTER-2 system was the first WYSIWYM-based system to use a textual user interface. Its predecessor, DRAFTER, used a graphical one.



DRIVE BELT REPLACEMENT

REMOVAL PROCEDURE

1. Remove the battery.
    a. Disconnect the negative battery cable.
    b. Disconnect the positive battery cable.
    c. Remove the retainer bolts.
2. Rotate the drive belt tensioner.
3. Remove the drive belt.
4. Remove the accessory drive belt from the engine.

INSTALLATION PROCEDURE

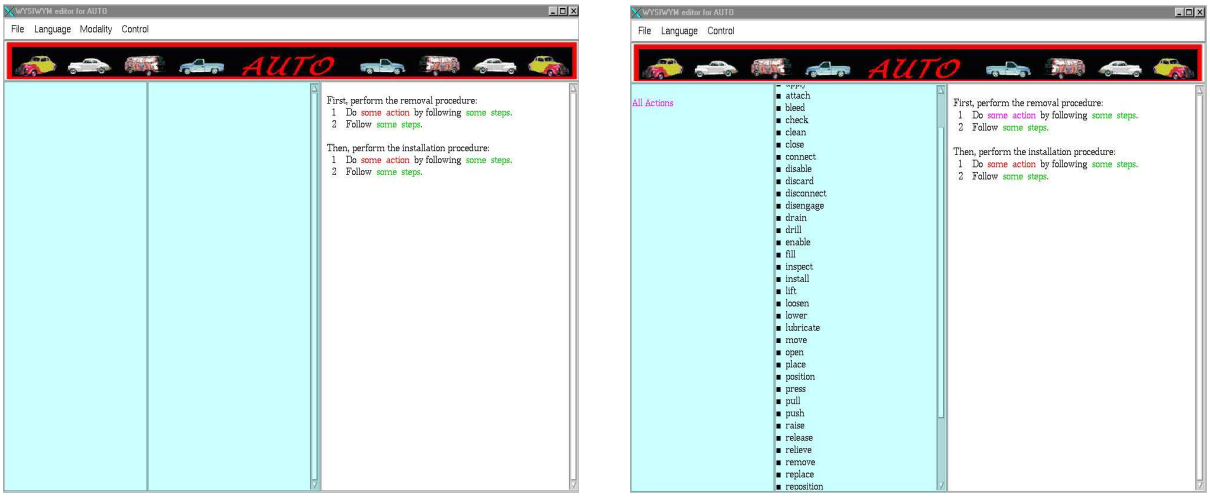
1. Position the accessory drive belt to the engine.
2. Rotate the drive belt tensioner.
3. Install the drive belt.
4. Install the battery.
    a. Install the retainer bolts.
    b. Connect the positive battery cable.
    c. Connect the negative battery cable.

Figure 2.1: Car Repair Procedure to Generate.

document production process from start to finish using AUTO. Section 2.2 describes the process used to obtain domain information for building AUTO. Section 2.3 provides an overview of the reference problems in current WYSIWYM-based systems this thesis solves.

2.1 AUTO: The Baseline WYSIWYM System for Authoring Car Repair Manuals

The WYSIWYM approach allows domain experts without previous exposure to computational linguistics or knowledge representation languages to construct a KB by interacting with a natural-language document in a menu-based knowledge editing system. Figure 2.1 shows a typical car repair procedure from the car repair domain



(a) (b)

Figure 2.2: User Interface for AUTO: Initially (a) and Following Selection of *Some Action* from Feedback Text (b).

that domain experts can use AUTO to generate. The initial text that AUTO presents to experts is shown in the screenshot of AUTO's user interface[2] in Figure 2.2(a). Domain experts interact with this natural-language text to create a KB for document production. This text, known as the "feedback text", communicates the current state of the KB and points out where it can be extended. Colored phrases indicate places where knowledge can be expanded. These "anchors" reflect unfilled slots in the KB. Red anchors, such as *some action*, represent arguments that are obligatory; they must be specified by the expert. Green anchors, such as *some steps*, signify arguments that are optional. A completely specified document can contain unspecified optional, but not obligatory arguments.

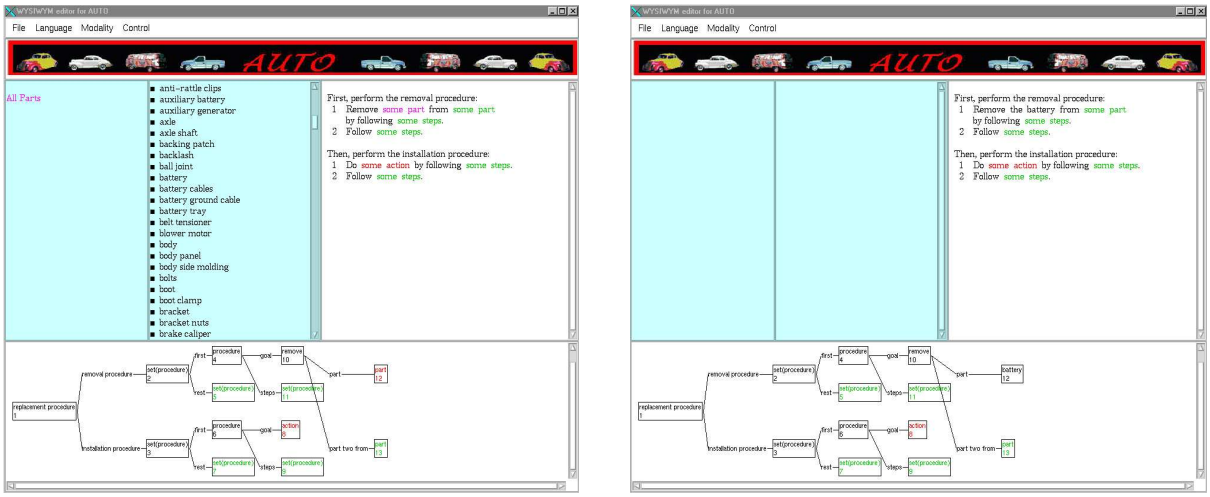
Figure 2.2(b) shows the result of clicking on the anchor *some action* in Step 1 of the removal procedure feedback text. AUTO presents a menu containing the fifty verbs

[2]The user interface for WYSIWYM-based systems is implemented in CLIM (CLIM 1994).

that represent the valid options for extending the KB at this point. Menu titles are displayed in the left-most pane, and the contents of the currently selected menu are shown in the middle pane. Pink is used in the menu title pane and the feedback text to indicate currently selected material. After the expert selects the verb *remove* from the action menu, the KB and feedback text are updated to reflect this choice. Figure 2.3(a) shows the resulting feedback text, along with a graphical representation of the updated KB.[3] The language-independent KB may be used to generate documents in multiple languages. It is shown in Figures 2.3 to provide a visualization of the representation that the system builds during the document production process. It is not used or seen by domain experts. The only interaction that experts have with the KB is mediated through the feedback text. The updated KB, together with the system's natural-language generation rules, are used to generate feedback text reflecting the updated KB. In Figure 2.3(a), the previously unspecified action in Step 1 has been replaced by the verb *remove*, which has two *part* arguments, only the first of which is obligatory. The expert specifies the obligatory *part* argument by selecting *battery* from the menu of 374 parts in the system's ontology, some of which can be seen in the figure. Figure 2.3(b) shows the updated KB and feedback text resulting from the expert's selection.

Figure 2.4(a) shows the feedback text after the document in Figure 2.1 has been completely specified using AUTO. Since all of the obligatory arguments have been specified, the expert has the option of viewing the "output text" by selecting *Output* from the *Modality* menu. The output text generated by the system, shown in Figure

[3]In description logic terminology, this representation is traditionally referred to as the assertional box, or a-box. It is a model containing specific facts about particular objects and relations in the domain.



(a) (b)

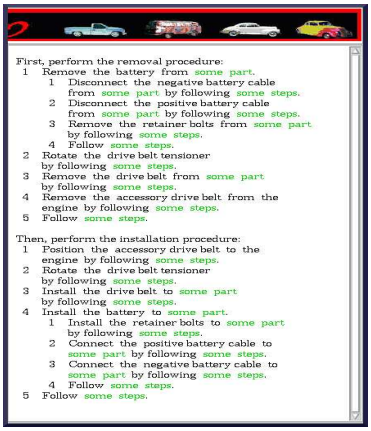
Figure 2.3: User Interface for AUTO with KB: Before (a) and After (b) *Battery* Part Selection from Menu.

2.4(b), is the intended final document, in this case the car repair manual text, and, therefore, does not include markings to indicate where knowledge may be added.

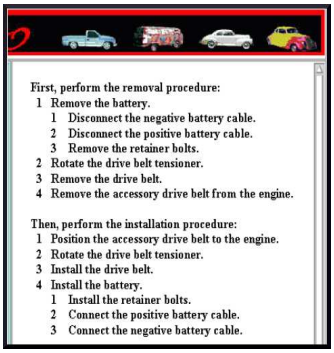
Figure 2.4(c) demonstrates the multilingual capabilities of WYSIWYM. The expert has chosen to generate the document in Spanish by selecting *Spanish* from the *Language* menu. Instead of using the English natural-language generation rules, the Spanish generation rules, along with the KB, are used to generate the output text in Spanish. In addition to generating output text in multiple languages, domain experts can also author documents in other languages.

2.2 Acquiring Domain Information for AUTO

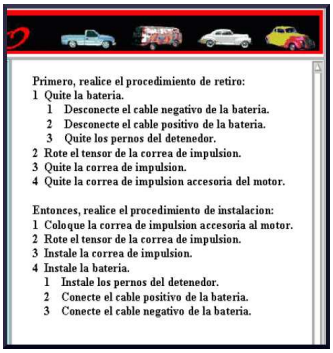
Building a WYSIWYM-based system for a particular domain requires three main components: an ontology, natural-language generation rules, and a document-structure



(a) English Feedback



(b) English Output



(c) Spanish Output

Figure 2.4: Text for Procedure in Figure 2.1.

model. To acquire this knowledge for the car repair domain, we analyzed a corpus of car repair manuals (General Motors 2003). Each manual consists of a number of car repair procedures, such as the one shown in Figure 2.1. The corpus is composed of approximately 600 car repair procedures containing about 8,500 instruction steps. The steps in the removal and installation procedures in Figure 2.1 are examples of instruction steps.

To create the ontology,[4] we automatically identified objects commonly occurring in car repair manuals and the relationships holding between them. Figure 2.5[5] illustrates the procedure we used:

1. Assign a part of speech tag to the words in each instruction step (Ratnaparkhi 1996).

[4] The ontology and natural-language generation rules for WYSIWYM-based systems are implemented in Prolog. In description logic terminology, the ontology is often referred to as the terminology box, or t-box.

[5] In this figure, # is a delimiter separating extracted verbs and noun phrases.

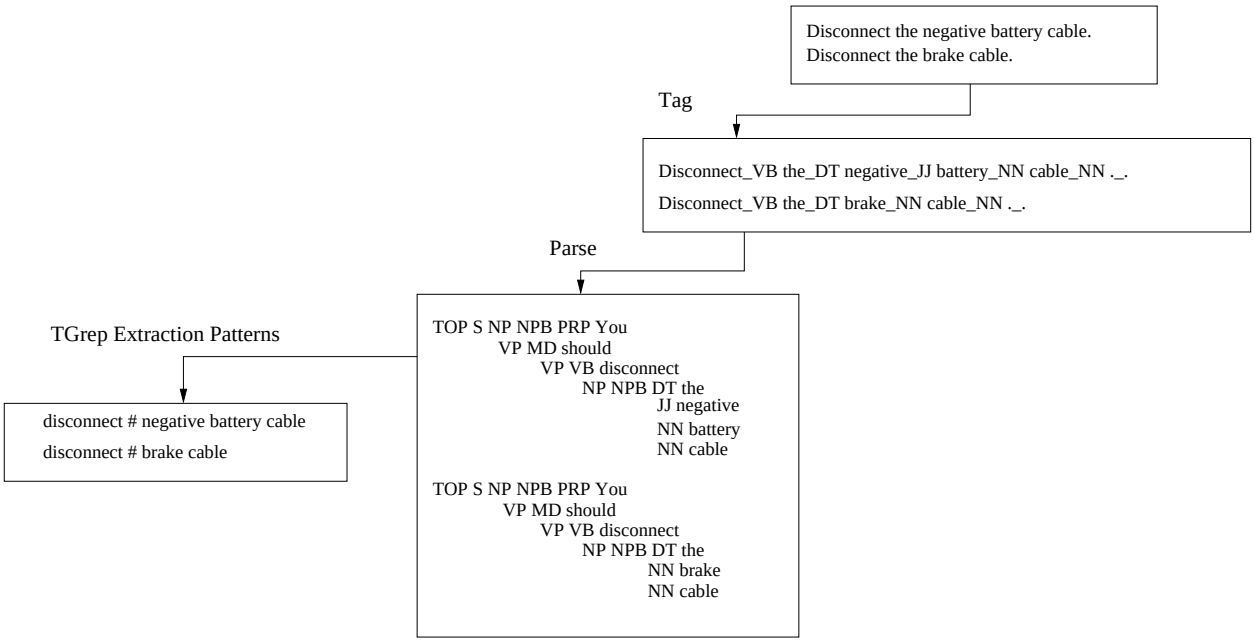


Figure 2.5: Ontology Creation.

2. Parse[6] each instruction step (Collins 1997).

3. Use syntactic patterns written in TGrep2[7] (Rohde 2001) to extract verbs and noun phrases from the instruction steps. This step resulted in the extraction of 8,725 verbs and 8,780 noun phrases.[8]

4. Include the fifty most frequently occurring verbs and all noun phrases (objects) extracted ten or more times (374 noun phrases) in AUTO's ontology.

[6]The tagger and parser were trained on the *Wall Street Journal*. We, therefore, introduce some errors that could be avoided if we had available manually tagged and parsed car repair manuals that could be used to retrain the tagger and parser. Also, since imperative sentences are, for the most part, nonexistent in the *Wall Street Journal*, we inserted the subject *you* and the modal verb *should* in all imperative sentences to improve the accuracy of the parser.

[7]TGrep2 is a software tool that, given a set of parse trees and a syntactic pattern using node names contained in the parse trees and relationships between these nodes, extracts those trees that match the pattern.

[8]These statistics reflect non-unique verbs and noun phrases.

5. Manually assign each object to a class. There are nine domain-specific classes, which include the following: part,[9] tool, defect, cleaning agent, fluid or adhesive, direction, position, pressure, and hole type.

To generate sentences about the objects in the ontology, the system requires syntactic information. For each argument structure of each verb, we created a natural-language generation rule. The argument structure for each verb consists of the number of arguments, their class, and whether each argument is obligatory or optional. This information was determined by manually inspecting parses containing the verb. A tool that automatically extracts verb subcategorization information from parses (Kinyon & Prolo 2002) could be used to automatically identify the number of arguments of a verb and whether each argument is optional or compulsory. This tool, however, requires that the parse trees accepted as input are annotated with "function tags" indicating semantic roles, which are not trivial to assign automatically (Blaheta & Charniak 2000).

The corpus analysis to uncover the global structural elements common to most car repair procedures, known as the document-structure model, revealed that they consist of two subprocedures: a removal procedure and an installation procedure. Each subprocedure consists of instructions steps. An instruction step is a single sentence that may be followed by an embedded subprocedure. The drive belt replacement procedure in Figure 2.1 exhibits the typical car repair procedure structure. It is composed of a removal procedure and an installation procedure. Step 1 of the removal procedure and Step 4 of the installation procedure are examples of instruction steps

[9]The *part* class contains the greatest number of objects.

Reference Problem	*Traditional WYSIWYM Method*	*New Method*	*Chapter*
KB Specification: New Entities	Alphabetical	Categorical Organization Rank: Domain and Lexical Knowledge	3
KB Specification: Already Mentioned Entities	Direct Manipulation	Rank: Discourse Structure and Genre Features	4
Document Generation	Full NP	Full or Reduced NP	5

Table 2.1: Improving the Naturalness of Reference in WYSIWYM-based Systems.

containing embedded procedures.

2.3 The Problem of Reference in WYSIWYM-based Systems

This thesis presents three new methods that individually and together make reference more natural in the KB specification and document generation phases of multilingual documentation production. Table 2.1 compares the new methods described in this thesis for dealing with reference to approaches used in traditional WYSIWYM-based systems.

In traditional WYSIWYM-based systems, menu options referring to new entities are organized alphabetically in flat lists, such as those in Figures 2.2(b) and 2.3(a). This thesis utilizes categorical organization for representing and communicating about entities. Within these categories of semantically similar options, domain and lexical knowledge is used to determine those entities to which domain experts are most likely to refer.

Most traditional WYSIWYM-based systems do not provide a means for domain experts to refer to entities that already have a representation in the KB. Those that do most often use direct manipulation. Figure 2.4 shows an example of a problem that may arise as a result of AUTO's inability to support repeated reference. The domain expert attempted to refer to several parts introduced in the removal procedure, such as the *engine* and the *battery*, again in the installation procedure. Because AUTO does not support coreference, however, the system assumed that each reference was to a new entity of the same type. This thesis describes mechanisms informed by the discourse structure of the current document and genre features of the task-specific car repair domain that allow experts to refer to already mentioned entities, and thus solves this problem.

Traditional WYSIWYM-based systems also do not deal with referring-expression generation. The referring expression used to refer to an entity in the feedback and output texts in Figure 2.4 is simply the noun phrase that was used to refer to the entity in the menu from which it was chosen. This may lead to documents that not only sound repetitive and unnatural, but are also misleading. For example, in Figure 2.4, the same referring expression (*the battery*) is used to refer to the battery in the removal procedure and the battery in the installation procedure, making it appear as if they are referring to a single battery when, in fact, they are referring to different entities that are both of type *battery*. This thesis presents a learned model informed by linguistic knowledge for generating full and reduced noun phrases. The resulting documents contain natural-sounding referring expressions clearly denoting the entities to which they refer.

Chapter 3

KB Specification: Referring to New Entities

3.1 Introduction

For a WYSIWYM-based system to work well, it must organize domain experts' choices in a manner that allows for quick and accurate selection. Previous WYSIWYM-based systems (Bouayad-Agha *et al.* 2002; Power, Scott, & Evans 1998, inter alia) organize options alphabetically in flat lists and disregard dependencies between objects already in the KB and objects that are potential candidates for selection. This chapter describes the creation of categories and the design of a statistical model that uses domain and lexical knowledge for organizing new options and presents an evaluation of AUTO, comparing a traditional knowledge-editing interface to the interface of systems informed by categorical organization and domain and lexical knowledge.

Section 3.2 describes the creation of categories and the statistical model for im-

posing order on the menus presented by AUTO. We created categories of objects by automatically examining the distribution of objects extracted from the corpus described in Chapter 2 and then grouped together those objects deemed to be related. To create the statistical model, we used the natural-language techniques of tagging, parsing, and extracting relevant syntactic arguments to automatically process the corpus and then collected statistics based on the frequencies of verb-noun phrase pairs. The model is used to produce "forecastable" menus containing new options (Nickerson 2003). Forecastable menus present entities that the statistical model indicates are likely to be specified in a given context based on verb-noun phrase pair frequencies. Use of categorical organization and the statistical model were, in part, motivated by the encouraged use of plagiarism in the car repair domain, i.e., reuse of text across years (Means & Godden 1996). The repetitive nature of the text allows us to group repeatedly mentioned entities into meaningful categories and to more accurately predict the contexts in which expressions referring to them are likely to occur. The four versions of AUTO, each employing a different method for organizing new options, developed to test the usefulness of our methods are presented in Section 3.3. Section 3.4 discusses the empirical evaluation conducted to determine the contribution of categorical organization and the statistical model that takes advantage of domain and lexical knowledge. Results from the study, presented in Section 3.5, show that categorical organization and the statistical model lead to shorter, structured menus from which subjects are able to quickly make selections. Feedback from the study motivated several improvements to the systems. Section 3.6 describes these improvements.

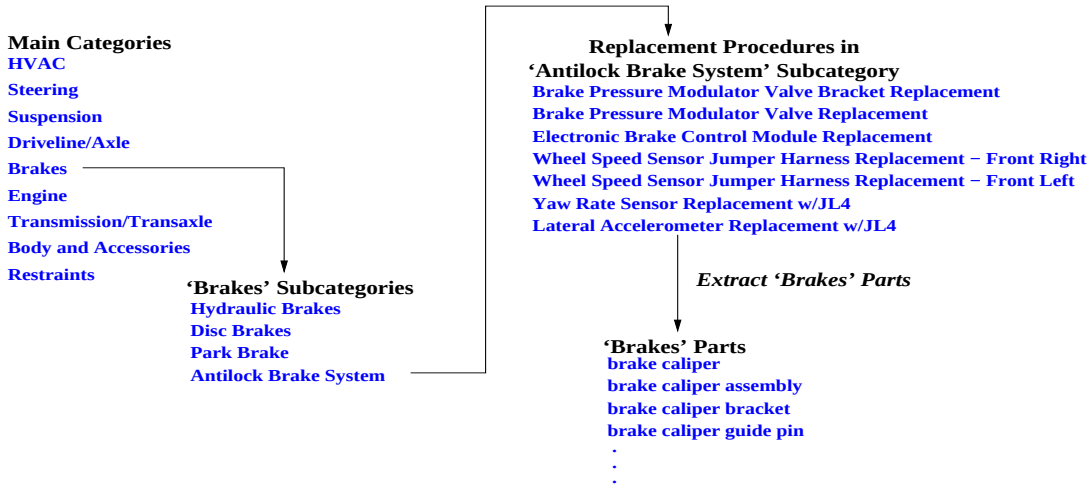


Figure 3.1: Construction of Car Part Categories.

3.2 Using Domain and Lexical Knowledge to Organize New Options

Parts is the class of objects that has the greatest number of members and the one whose members appear most frequently in car repair manuals. To better organize the lengthy list of car parts, we placed them into nine categories based on information extracted automatically from General Motors' repository of car repair information (General Motors 2003). The categories are: HVAC (heating, ventilation, and air conditioning), steering, suspension, driveline/axle, brakes, engine, transmission/transaxle, body and accessories, and restraints.

The organization of the nine main categories of car repair procedures in the General Motors' repository is depicted in Figure 3.1. These categories are linked to subcategories which are, in turn, linked to car repair procedures. To determine category membership for each extracted car part from Chapter 2, we determined the main cat-

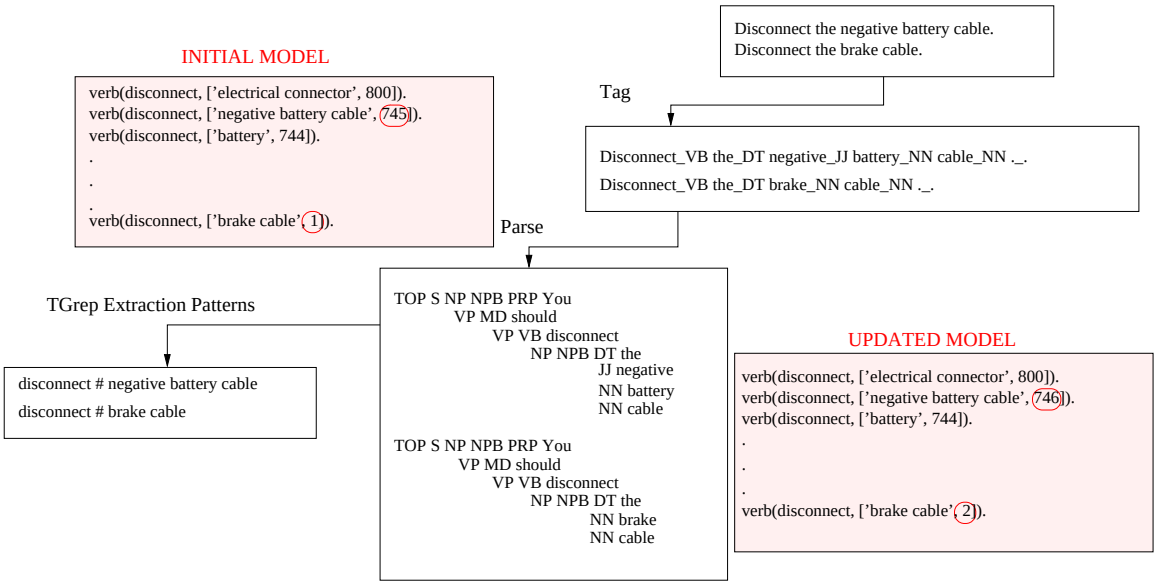


Figure 3.2: Development of the Statistical Model.

egory of the car repair procedure containing the part. The car part was then placed in this category. For example, all of the car parts extracted from the *Brake Pressure Modulator Valve Bracket Replacement* procedure are members of the category *Brakes* since this procedure is a member of the *Antilock Brake System* subcategory of the *Brakes* main category. A part can be a member of more than one category.

We developed a statistical model that uses local context as leverage to further organize new options. To create the model, we processed the corpus of car repair manuals using natural-language techniques and then computed statistics based on the frequency of verb-noun phrase cooccurrence patterns. The frequency counts of the final model can be used to construct forecastable menus that predict which objects are most likely given the specified verb for the current instruction step.

To develop the statistical model, we used a procedure similar to the one used to create the ontology for the car repair domain. Figure 3.2 shows how the update

procedure works for the statistical model. The procedure associates the noun phrases extracted from each instruction step with the main verb of the instruction step. Each time a given verb-noun phrase pair is observed, the pair's current frequency count is incremented. In the figure, for example, the car parts *negative battery cable* and *brake cable* are observed in the context of the verb *disconnect*. This results in the incrementation of the frequency counts (which have been circled in red) for the pairs disconnect:negative battery cable and disconnect:brake cable. This process continues until all of the instruction steps in the corpus have been processed. The cooccurrence statistics we collect are similar to those used by others (Kehler *et al.* 2004; Dagan & Itai 1990) who use verb-noun phrase frequencies observed in a corpus to organize the possible referents of an anaphor.

The statistical model uses semantic dependencies acquired automatically and validated empirically. Prior work (Brun, Dymetman, & Lux 2000) makes use of manually specified and, therefore, less extensible object dependencies. The dependencies used in previous work are employed to present options in "context-aware" menus. They help to restrict objects that can be chosen to extend a KB. For example, choosing the verb *swallow* restricts the object of the verb to include only those objects that are capable of being swallowed.

3.3 Systems that Rank New Options

To determine the contribution of categorical organization and the statistical model, we created three versions of AUTO, each of which supports a different method for organizing new options in menus presented to experts as they edit a KB. The first

version uses categorical organization, the second uses the statistical ordering, and the third uses both categorical organization and the statistical ordering. To demonstrate the usefulness of the different methods, we compared them to the baseline system that organizes new options alphabetically, the manner used by most traditional knowledge-editing systems. An example menu produced by each version when *some part*, in the context of the verb *connect*, is being specified is shown in Figure 3.3. A more detailed description of each system follows:

1. **AUTO** (Figure 3.3(a)), like prior work, simply alphabetizes all car part objects contained in the system's ontology and presents them in a single menu containing 374 options.

2. **AUTO+cat** (Figure 3.3(b)) was created using the nine car part categories that were induced from the structure of the car repair information repository. When choosing the value for a car part argument, domain experts can view, separately, nine lists, one for each car part category. If the car part experts are searching for has been previously encountered in a procedure in the same main category as the one they are currently specifying, the part will be contained in the appropriate category menu. In Figure 3.3(b), the *All Engine Parts* menu containing 226 options is shown. Engine parts are the type of part most often occurring after the verb *connect*. In the left menu pane, titles of the nine different car part category lists appear. The right menu pane is a result of clicking on the menu title *All Engine Parts*.

3. **AUTO+vo** (Figure 3.3(c)) consults the statistical model to organize menus. This system produces two menus for each car part that experts specify: one

containing the common car parts in a given context, and the other, all car parts. Once domain experts choose a verb, AUTO+vo organizes valid objects of the verb based on the number of times they occurred with the verb in the corpus. Verb-noun phrase pairs are used as selection patterns to give preference to certain new options given the context, namely the preceding verb, in which they are to appear. Those objects that occurred more than five times with the preceding verb in the corpus are presented in a separate *Common Parts* menu. In this example, there are 33 objects that are *common* after the verb *connect*. The second menu simply contains all 374 car parts in the ontology–this menu is identical to the one presented by AUTO.

4. **AUTO+cat+vo** (Figure 3.3(d)) is a system that takes advantage of categorical organization and the statistical model. This system contains at most eighteen different menu titles in its left menu pane. The *common* menu titles (at most nine) are ordered based on the number of times that a part of each type was observed with the verb of the instruction step currently being specified. In the example in Figure 3.3(d), for instance, engine parts are observed more frequently with the verb *connect* than any other part type. The right menu pane shows the *Common Engine Parts* menu presented by AUTO+cat+vo. The menu contains 26 engine part options that commonly occur after the verb *connect* (compared to 374 options presented by AUTO in (a), 226 options presented by AUTO+cat in (b), and 33 options presented by AUTO+vo in (c)).

Figure 3.3: Menu Organization for Specifying *Some Part* in the Context of the Verb *Connect*.

3.4 Experimental Design

The main goal in evaluating AUTO is to determine if subjects find the organization imposed by the car part categories and the statistical model useful. We compared the four systems for authoring car repair procedures discussed in Section 3.3: AUTO, AUTO+cat, AUTO+vo, and AUTO+cat+vo. AUTO, like many existing knowledge-editing systems (Power, Scott, & Evans 1998; Bouayad-Agha *et al.* 2002, inter alia), organizes options using an uninformed method, namely alphabetically. The other three systems use innovative, informed methods that employ categorical organization and a statistical model based on domain and lexical knowledge. Comparing AUTO to AUTO+cat and AUTO+vo to AUTO+cat+vo tests the contribution of categorical organization and comparing AUTO to AUTO+vo and AUTO+cat to AUTO+cat+vo highlights the contribution of the statistical model.

The following three factors were controlled in the study:

1. **System**: All subjects interacted with four systems: AUTO, AUTO+cat, AUTO+vo, and AUTO+cat+vo. None of the systems supported *copy* and *paste* editing operations. The systems did, however, have available *undo* and *cut* operations for correcting mistakes.

2. **Procedure Set**: We divided eight car repair procedures chosen randomly from a held out set[1] of procedures into four procedures sets, each containing two procedures. Each subject authored all eight repair procedures, two per system. Each procedure contained removal and installation subprocedures. The struc-

[1]None of the procedures were part of the collection used to determine categorical membership or to create the statistical model.

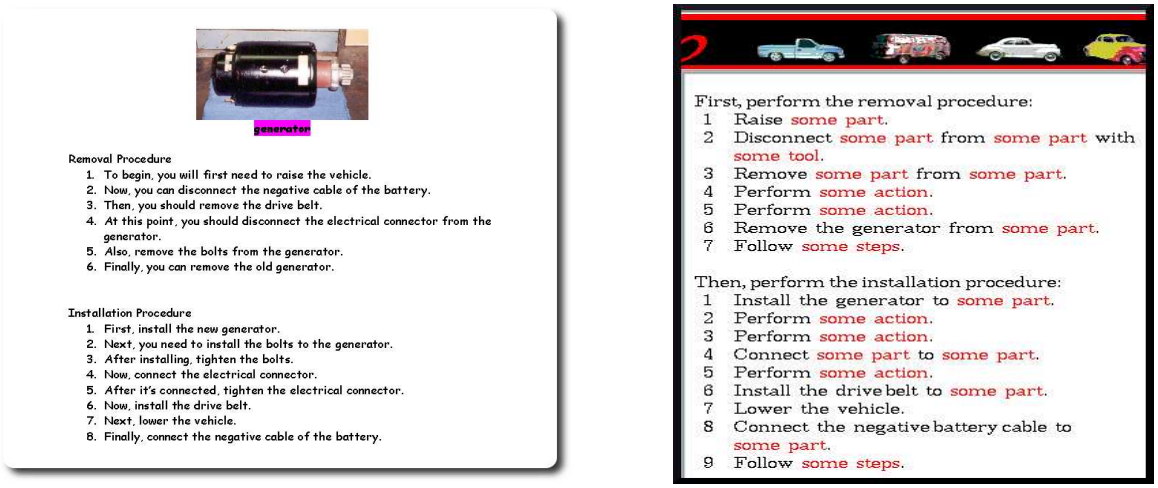
ture of each procedure was flat, i.e., none of the instruction steps contained embedded subprocedures. All anchors marking where knowledge could be specified were *red*, however, subjects were told that they did not need to specify a value for all of the anchors. Subjects were told to assume that the system would be able to decipher cases in which they had to refer to the same object again; they did not have to explicitly specify this type of reference. An example repair procedure description is shown in Figure 3.4(a). Subjects were given the template in Figure 3.4(b) as a starting point. They were told that they may not be able to reproduce the procedure description word for word. Instead, the description was meant to equip them with the mechanical knowledge necessary to author a procedure that a repair-person would be able to follow.

3. **Order**: For the experiment, subjects were randomly divided into four groups of four subjects. To compensate for carryover effects, i.e., learning due to presentation order, a balanced Latin Square[2] (Martin 2003) determined the order in which subjects interacted with the systems. In addition to counterbalancing system interaction order, within each group of subjects, a balanced Latin Square also dictated the order in which subjects authored the four procedure sets.

Sixteen subjects[3] interacted with each system to author two car repair procedures per system. They filled out the short questionnaire in Appendix A to assess user

[2]A balanced Latin Square is one in which, in addition to each system appearing with equal frequency in each position, each system also appears before and after each other system an equal number of times.

[3]The subjects were graduate students who had no experience using AUTO. They were, however, given a tutorial prior to using each system. The tutorial they were given before using AUTO+vo can be found in Appendix A.



(a) (b)

Figure 3.4: Description (a) and Template (b) for the Generator (an *Engine* Part).

satisfaction after interacting with *each* system. For each subject/system/procedure set triplet, the system logged selection time,[4] the elapsed time between when subjects chose a menu title and when they made a selection from the list of displayed options, for each verb and car part option.[5] For each of the four systems, the experiment resulted in the authoring of 32 procedures and the collection of 424 selection times (144 for verbs and 280 for car parts).

[4]Subjects were not explicitly told they were being timed, but they were encouraged to "write procedures as fast as possible while maintaining accuracy." Also, they were made aware that they had a time limit for using each system to author the two procedures they were given.

[5]After clicking on an anchor in the feedback text, subjects were presented with a list of menu titles in the left-most menu pane. They had to click on a menu title before they were able to view possible verb and car part selections. The systems did not present a list of options initially.

3.5 Results and Discussion

We determined the contribution of categorical organization and the statistical model by evaluating the fulfillment of three conditions: menu **containment**, correct menu **examination**, and menu **length**. The fulfillment of these evaluation conditions led to a speed-up in selection time[6] when categorical organization and the statistical model were used, compared to when menu options were alphabetized in a flat list. Overall, subjects authored car repair procedures most efficiently when they interacted with systems[7] that presented shorter menus containing the desired options.

3.5.1 Contribution of Categorical Organization

For categorical organization to be beneficial, it must lead to a speed-up in selection time compared to systems that do not use categorical organization. For this to be possible, the three evaluation conditions must be satisfied: 1) The car part options were contained in the appropriate category menu (**containment**). 2) Subjects were able to determine the correct category menu to examine (**examination**). 3) The category menus contained significantly fewer options than the menus of systems that did not use categories (**length**).

The **containment** condition was satisfied for 100% of the car parts the subjects had to specify. All of the parts were contained in the category from which the car

[6]Task success is a feature that we do not analyze since the procedures proved to be sufficiently simple for subjects. All of the subjects were able to author correct procedures using all of the systems.

[7]Because we counterbalanced both the order in which subjects authored procedure sets and system interaction order, we assume no interaction effects among the controlled factors **procedure set** and **order**. Further, effects based on **procedure set** are not insightful because individual procedures contain varying numbers of instruction steps. The analysis in this chapter focuses on comparisons between systems.

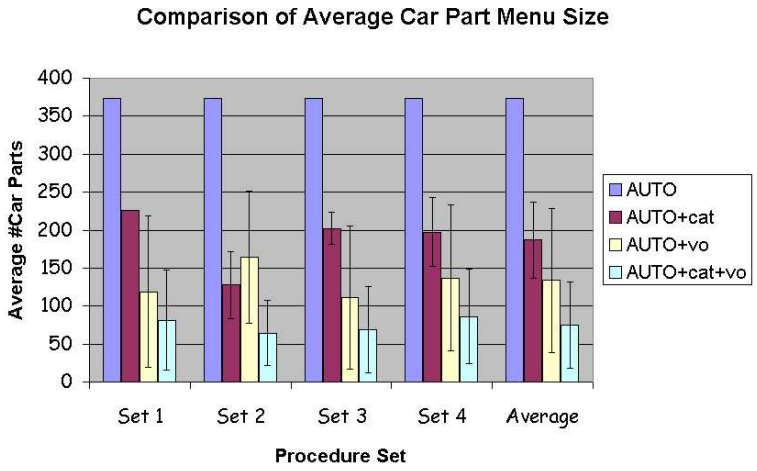


Figure 3.5: Average Car Part Menu Size (in Number of Options) for Four Versions of AUTO.

repair procedure the subjects had to specify belonged. For instance, all parts that had to be specified in the car repair procedure for the battery, an engine part, could be found in the *All Engine Parts* category menu.

The **examination** and **length** conditions were partially satisfied. Subjects indicated that, at first, they did not know where to look for the appropriate part, but once they discovered which menu contained the parts they were searching for, they were able to locate them quickly. Overall, categorical organization allows AUTO+cat and AUTO+cat+vo to present category menus containing fewer options when compared to the menus presented by AUTO and AUTO+vo, respectively. Figure 3.5 shows a comparison of the average number of menu options per procedure set that each system presented.[8] The category menus presented by AUTO+cat contained an average of 187 options, half the number of options in the menus presented by AUTO. AUTO+cat+vo presented category menus containing an average of 75 common op-

[8]Only those car part anchors left unspecified initially in the procedure templates are accounted for in Figure 3.5.

Category	*#Car Parts*	*Reduction from All 374 Car Parts*
Restraints	62	83.4%
Driveline	85	77.3%
Suspension	95	74.6%
Steering	109	70.9%
Brakes	123	67.1%
HVAC	124	66.8%
Transmission	129	65.5%
Body	184	50.8%
Engine	226	39.6%
Average	126 $\pm$ 50	66.2% $\pm$ 13.5%

Table 3.1: #Car Parts in Each of the Nine Car Part Categories: **Length** Condition.

tions, 44% fewer common options than the menus presented by AUTO+vo. Table 3.1 shows the number of parts contained in each of the nine categories. The wide range in the number of car parts contained in the categories explains the large standard deviation in Figure 3.5 for systems making use of categorical organization. Even though for some categories, the *engine* category for instance, the reduction in the number of car parts is not very drastic (around 40%), for other categories, the number of options is greatly reduced, by as much as 83% for the *restraints* category. For specifying the final three[9] car parts in the car repair procedure for the air tube, a *suspension* part, the average selection time using AUTO+cat was 6.5 $\pm$ 0.3 seconds, compared to 6.9 $\pm$ 0.3 seconds for the final three parts in the car repair procedure for the generator, an *engine* part.

Table 3.2 show the average selection time for car parts and verbs for the four systems. The fulfillment of the **containment** condition and the partial fulfillment of

[9]We examined the final three parts because by this time, subjects had determined the correct category menu to use.

Object Type	*AUTO*	*AUTO+cat*	*AUTO+vo*	*AUTO+cat+vo*
car parts	7.5 ± 0.6	6.8 ± 0.4	4.9 ± 0.9	3.7 ± 0.3
verbs	3.3 ± 0.3	3.4 ± 0.2	2.3 ± 0.2	2.1 ± 0.2

Table 3.2: Average Selection Time (in Seconds) for Car Parts and Verbs Specified Using Four Versions of AUTO.

the **examination** and **length** conditions explains why, when comparing AUTO+cat to AUTO ($p < 0.04$) and AUTO+cat+vo to AUTO+vo ($p < 0.008$), the reduction in car part selection time is significant.[10] Differences in selection time are most pronounced when the car part category greatly reduced the menu length. In summary, categorical organization produces an intuitive grouping of car parts and shorter menus from which subjects were able to make selections more efficiently, the degree to which depended on the number of car parts contained in the category.

3.5.2 Contribution of the Statistical Model

For the statistical model to be useful, as with categorical organization, it must lead to a speed-up in selection time. The three evaluation conditions restated in terms of the statistical model are as follows: 1) The verb and car part options subjects were looking for were contained in the menus created as a result of consulting the statistical model (**containment**). 2) Subjects initially examined the *common* menus as opposed to using the *all* menus that contained all possible selections (**examination**). 3) The *common* menus were significantly shorter than the *all* menus (**length**).

Table 3.3, which displays the number of objects the statistical model predicted to be *common*, shows that the **containment** condition was satisfied. For 30 of the

[10] We compare pairs of system versions using a paired sample t test. $p < 0.5$ is considered statistically significant (Cohen 1995).

Procedure Set	*#Common Car Parts/ #Car Parts Total*	*#Common Verbs/ #Verbs Total*
1	16/18	6/9
2	12/19	8/8
3	13/16	8/9
4	16/17	8/10
Total	57/70 (81.4%)	30/36 (83.3%)

Table 3.3: Objects Predicted to be *Common* by the Statistical Model Used by AUTO+vo and AUTO+cat+vo: **Containment** Condition.

36 verbs (approximately 83%), the verb was contained in the *Common Actions* menu used by AUTO+vo and AUTO+cat+vo.[11] For 57 of the 70 car parts (around 81%), the statistical model predicted that the car part subjects added to the KB was likely to occur with the verb of which it was an object. In other words, for 81% of the car parts that subjects had to specify, the part could be found in the shorter *Common Parts* menu that AUTO+vo presented. For the *common* menus to be helpful in AUTO+cat+vo, subjects first had to correctly identify the type of part that they wished to add to the KB. Once the part's category was correctly identified, 81% of the parts being specified were found in the *common* menu for this car part category.

The **examination** condition was also met. Subjects first examined *Common Actions* menus presented by AUTO+vo and AUTO+cat+vo 97% of the time for verbs. Subjects first examined the *Common Parts* menus used by AUTO+vo 92% of the time, and they initially consulted a *common* car part category menu 86% of the time when using AUTO+cat+vo.

The **length** condition was satisfied for verbs. *Common Actions* menus contained

[11] The *Common Actions* menus presented by AUTO+vo and AUTO+cat+vo were identical. They contained the ten most frequently observed verbs in the background corpus. The only difference in the menus presented by AUTO+vo and AUTO+cat+vo is the categorical organization that the latter imposes when presenting menus of car parts.

ten verbs, compared to the 45 in the *All Actions* menus. The fulfillment of these three conditions for verbs, therefore, explains the speed-up in selection time for verbs in Table 3.2 when comparing AUTO and AUTO+cat to the two systems that use the *Common Actions* menu, AUTO+vo ($p \approx 0$) and AUTO+cat+vo ($p \approx 0$). The **length** condition was partially met for car parts. The statistical model allows AUTO+vo and AUTO+cat+vo to present *common* menus containing fewer options than the menus presented by AUTO and AUTO+cat, respectively. The graph in Figure 3.5 shows that all car part menus presented by AUTO contained all 374 of the car parts. The menus that AUTO+vo presented, however, contained an average of 134 options, 240 fewer options than the menus presented by AUTO. The menus that AUTO+cat constructed for the nine different categories of car parts contained an average of 187 options. When AUTO+cat+vo consulted the statistical model to create menus, the average number of options fell to 75 options, a reduction of 112 options. Due to this reduction in menu length, it is not surprising that in Table 3.2, when comparing AUTO and AUTO+cat to AUTO+vo ($p \approx 0$) and AUTO+cat+vo ($p \approx 0$), respectively, there is a significant speed-up, overall, in selection time for car parts.

Table 3.4, which presents all verbs that subjects had to specify in the procedures they authored and the number of common car parts that occur with each verb, however, presents a more detailed picture of the statistical model's performance. The table reveals that some verbs occur frequently with only a small number of objects. These informative verbs, such as *tighten*, have a high information content and impose selectional restrictions on the objects most likely to follow them. For example, only a small number of objects such as *bolts* and *fasteners* can be tightened. Other

Verb	*#Common Car Parts*	*Reduction from All 374 Car Parts*
raise	10	97.3%
rotate	10	97.3%
tighten	10	97.3%
lower	11	97.1%
position	26	93.0%
connect	33	91.2%
disconnect	33	91.2%
install	211	43.6%
remove	218	41.7%
Average	62 $\pm$ 87	83.4% $\pm$ 23.2%

Table 3.4: *Common Parts* Menus for Verbs in AUTO+vo: **Length** Condition.

verbs, such as *remove*, occur frequently with most objects in the domain of car repair procedures. These uninformative verbs[12] have a low information content. They are used frequently with most objects; therefore, the statistical model is unable to greatly restrict the objects likely to occur with them. Table 3.5 presents average selection times using AUTO+vo for specifying car part objects following verbs in different informativeness classes. Informative verbs include *raise*, *rotate*, *tighten*, and *lower*. Members of the semi-informative verb class are *connect*, *disconnect*, and *position*. *Install* and *remove* make up the uninformative verbs. The table shows that the average selection time for car parts occurring with informative verbs in AUTO+vo is 3.3 seconds, which is lower than the average selection time for the system overall. The average selection time for uninformative verbs is 6.6 seconds, much closer to the average selection time when the subjects were presented with the entire list of car parts. To sum up, subjects found the information provided by the statistical model

[12] *Is* and *has* are canonical examples of uninformative verbs.

Verb Informativeness	*AUTO+vo*
Informative	3.3 ± 0.2
Semi-Informative	3.6 ± 0.3
Uninformative	6.6 ± 0.2

Table 3.5: Average Selection Time (in Seconds) for Car Parts Following Verbs of Varying Informativeness in AUTO+vo.

helpful for specifying verbs. For car part specification, they found the knowledge the system gained from the statistical model most useful when the information led to a significant reduction in the size of the *common* menu, i.e., when the verb in the local context was informative and, therefore, imposed selectional restrictions on the objects most likely to follow it.

The analysis of the two informed methods for organizing new options reveals that a system that uses either categorical organization or the statistical model for organizing new options leads to more efficient car repair procedure specification when compared to a system that uses alphabetical organization, the uninformed method employed by traditional knowledge-editing systems. Further, a significant speed-up in selection time was observed in systems using both categorical organization and the statistical model when compared to systems using only one of the informed methods. Even though the speed-up in selection time for individual verbs and car parts is, in some cases, no more than a second, a car repair procedure contains an average of ten verbs and ten car parts. Car repair manuals, in turn, are made up of many car repair procedures. A system that makes use of informed organization methods, therefore, could lead to improved productivity and efficiency among domain experts.

3.6 System Improvements Based on User Study Feedback

An analysis of the questionnaires filled out by subjects who participated in the study suggested three improvements to the systems. Two improvements were a result of subjects being displeased with the number of menu selections they were forced to make. There were two cases in which the number of selections could be reduced. The first included insertion and deletion operations in which subjects were asked to make a selection from a menu containing only one option. For insertion operations containing a single option menu in which subjects had to explicitly choose the single option, the systems were modified to automatically extend the KB to include the single valid menu selection. For cut operations with single option menus, on the other hand, the systems remained unchanged; subjects have to explicitly choose the single menu selection. This safeguard prevents subjects from inadvertently cutting large portions of the currently specified KB.

The systems in the study also forced subjects to choose which list of options to view first. For instance, when subjects chose to expand an action anchor, they were required to choose whether to view all of the verbs or common verbs first. The improved systems determine which menu is most likely to be viewed by the subjects and then present this menu initially. For AUTO, there is only one menu option, the *all* option, so all of the options are presented initially. AUTO+vo has two menu options, *common* and *all*. Based on the results of the study, subjects most frequently viewed the common menus first, and these menus led to a reduction in option selection time.

Figure 3.6: Text Generated to Reflect Extension of KB to Include Car Part Being Replaced.

Therefore, for AUTO+vo, it is the options in the common menus that are presented initially.

AUTO+cat and AUTO+cat+vo make use of the nine car part categories discussed in Chapter 3. To determine which car part category menu should be presented initially, we redesigned the KB to include an object for the type of car part for which subjects are writing repair procedures. The text to represent this KB extension is presented in Figure 3.6. Subjects are now encouraged to specify the car part before writing the replacement procedure for it. The system then uses the selected part to determine which menu to present initially. For example, if subjects indicate that they intend to write a replacement procedure for a generator, a member of the engine category, whenever they choose to specify a car part anchor, they are initially presented with the list of all engine parts when interacting with AUTO+cat and a list of all common engine parts when using AUTO+cat+vo.

3.7 Possible Extensions

There are a number of avenues of future research for the work presented in this chapter: refining the design of categories, improving the accuracy of the natural-language tools used to develop the statistical model, and enhancing the developed model. The method for creating categories of car parts presented in this chapter is only a first approximation at creating groups of similar objects. A more principled method for constructing categories would involve assembling a pool of experts and asking them to categorize the domain objects. For the car repair domain, for instance, we could ask people with mechanical knowledge to place each car part in a single category based on their knowledge of how car components function, given the nine main car part categories. Our method for determining category membership resulted in much inter-category overlap. A panel of experts, however, would likely be able to assign a given object to a single best category based on knowledge and experience.

The tagger and parser were trained on sentences from the *Wall Street Journal*. It would be interesting to retrain these tools on sentences more similar to those found in car repair manuals. A parser and tagger more attune to the terminology and syntactic structures encountered in the car repair domain would lead to more reliable parses, thus allowing car parts to be extracted more accurately. A statistical model developed using natural-language tools customized to the car repair domain would make more accurate predictions. This endeavor, however, would require a tagged and parsed corpus of car repair manuals, which is not currently available.

There are several ways in which the statistical model could be extended and improved. The system could examine manuals already generated by domain experts

and the manual currently being specified to update the statistical model online. If these sources of information were consulted, even though the corpus might not contain manuals similar to those experts have authored, as they continued to author more manuals, the statistical model would become tailored to their use of objects and organize new options to more accurately reflect their usage of the ontology.

Verbs for which little data are available could be leveraged by verbs occurring with the same objects for which significant data are available. For example, not many objects occur frequently with the verb *discard* in the corpus, but this verb is similar, syntactically and semantically,[13] to the verb *remove*, with which many objects occur frequently. Cooccurrence statistics collected for *remove* could be used to predict those objects most likely to be associated with the verb *discard*.

The statistical model could be used to inform a dialogue interaction between the system and domain experts. If the system had enough evidence regarding the option most likely to be chosen, the system could take initiative and make decisions for experts. In a more likely scenario, the system could propose a few of the most likely options to experts; and then, if experts did not wish to select any of these options, the system could resort to presenting a more complete list. If the system did not have enough information regarding the most likely options in a given context, the system could default to presenting experts with a complete list of options.

Feedback from the user study indicated several alternative, but related approaches for revising the *common* menus used by AUTO+vo and AUTO+cat+vo.

1. Include only the ten most frequent objects occurring more than five times in

[13]More precisely, the verbs have an argument structure in common and have a similar meaning.

the background corpus with a given verb.

2. Construct *common* menus by first defining a range of the number of options to include in them, for example 10-15. Then, examine a list containing the frequency counts of the objects with the 10^{th}-15^{th} highest frequency counts in descending order. Impose a cutoff between the two consecutive counts in the list whose difference is the greatest. Then, include all elements in the *common* list with a frequency greater than or equal to the larger of these two counts.

3. Use mutual information (Manning & Schutze 1999) to compute the extent to which a verb determines an object. The *common* menu would contain only those objects most affected by the value of the verb, not those objects with mutual information scores close to zero.[14]

4. Only associate *common* menus with arguments of verbs that are informative. This approach follows from the fact that including fewer items in the *common* menu (see approaches 1-3 above) would lead to the parts no longer being found in the *common* menus when a verb is uninformative. To avoid forcing subjects to decipher which verbs are the uninformative ones for which examining the *common* menu is not fruitful, the system would do this automatically.

In addition to demonstrating the usability of a statistical model that uses domain and lexical knowledge to organize new options represented by linguistic expressions, it would be interesting to demonstrate the usefulness of such a model in organizing options in other types of symbolic authoring interfaces. For example, a statistical

[14] A mutual information score of zero indicates that the two variables are independent.

model could be used to organize new options in graphical browsers (Skuce & Lethbridge 1995) given a background corpus of diagrams from the appropriate domain.

Chapter 4

KB Specification: Referring to Already Mentioned Entities

4.1 Introduction

In addition to referring to new entities (cf. Chapter 3), domain experts may also need to refer to objects that already have a representation in the KB. This linguistic phenomenon, known as coreference,[1] is an important problem in symbolic authoring approaches to multilingual document production. If the system does not recognize cases of coreference, the KB will be inaccurate. For instance, it might contain multiple KB objects for the same entity. Since a correct KB is a prerequisite for generating text with correct referential forms, it is essential that the system and experts agree

[1] This chapter focuses on the type of coreference in which noun phrases that are coreferential take part in an identity relation, i.e., they have the same referent. This is the most common type of coreference in the domain of car repair procedures. For the generation of more complicated texts, other semantic relations, such as bound anaphora and association or bridging reference (Clark 1977), may also need to be taken into account (Poesio, Bruneseauz, & Romary 1999; van Deemter 1992, inter alia).

on the entities to which are being referred. In the examples so far, we have ignored cases of coreference. For instance, when subjects referred to the *generator* object to specify Steps 4 and 5 of the removal procedure in Figure 3.4(a), they assumed the system would be able to infer that they were referring to the same *generator* entity in both steps. Usually, however, resolving cases of coreference is not this easy, and the assumption that the system can correctly determine cases of coreference is an unrealistic one.

Knowledge representation languages used to specify a KB in traditional MNLG approaches have operators that allow a knowledge engineer to encode instances of coreference explicitly. For example, in the CLASSIC language (Borgida *et al.* 1989), the "same-as" operator is used to indicate that two instances of the same type share the same referent. Coreference specification in symbolic authoring approaches is more challenging, especially for those with textual interfaces.[2] As pointed out in Chapter 2, early WYSIWYM-based systems ignore coreference all together. In DRAFTER-2 (Power, Scott, & Evans 1998), experts have no control over coreference. When experts attempt to refer again to an entity, the system creates a new object in the KB. ICONOCLAST (Bouayad-Agha *et al.* 2002), a system for authoring patient information leaflets, and AGILE (Hana 2001), a system for writing computer aided design manuals, allow domain experts some control over coreference. To specify a coreference relation, experts copy the text referring to the desired antecedent and then paste it in the new context where they wish to refer to it again. DRAFTER-3 (van Deemter & Power 1998), the successor of DRAFTER-2, addressed some of

[2]Human-computer interfaces for a variety of applications, such as building parse trees (Bird *et al.* 2002) and family trees (GenDesigner 2004), must also address the problem of coreference.

the shortcoming of previous WYSIWYM-based systems with regard to coreference. Specifically, if experts are about to specify an object at a location where coreference is possible, the system queries them to determine if they wish to refer to an existing KB object. DRAFTER-3, however, was a proof-of-concept implementation for a small-scale application whose purpose was to demonstrate that the ability to refer to already mentioned entities was a necessary requirement for KB specification. The system did not investigate different methods for referring to already mentioned entities during KB specification, nor did it provide empirical evidence to demonstrate the effectiveness of the single method it employed.

This chapter focuses on extensions to AUTO that enable the system to communicate with domain experts to resolve potential coreference relations. This task differs from the traditional coreference resolution task (Baldwin 1997; Ng & Cardie 2002, inter alia). Coreference resolution in the traditional sense requires that a system, using only a text, determine which expressions refer to the same entity. In AUTO, experts and the system must also agree on which expressions have the same referent, but some of the burden shifts to experts; the system, having identified a potential coreference relation, proposes coreference candidates and then solicits feedback from the experts to determine if they wish to refer to one of these entities again.

This thesis presents innovative mechanisms for controlling linguistic coreference in knowledge-editing systems (Nickerson 2002). The mechanisms vary according to the following four properties:

1. Policy type for locating potential coreference relations and candidates [**policy type**] (Section 4.2);

2. Factors, such as discourse structure and genre features, used to rank coreference candidates [**ranking algorithm**] (Section 4.3);

3. Mechanism used for reaching a consensus between the expert and the system with regard to coreference relations [**communication style**] (Section 4.4); and

4. Representation of the coreference relations in the user interface [**UI content**] (Section 4.5).

The mechanisms incorporate novel methods for locating potential coreference relations, ranking coreference candidates, and representing specified coreference relations. This chapter also presents an evaluation of AUTO extended to support coreference, comparing the alphabetical organization of coreference candidates, the organization method employed by most knowledge-editing interfaces that handle coreference, to systems that use information provided by the discourse structure and genre features to organize coreference candidates. We created three versions of AUTO, described in Section 4.6, that use different methods for organizing already mentioned options. Section 4.7 discusses the empirical evaluation designed to highlight the contribution of taking discourse structure and genre features into account when ranking coreference candidates. The results of the evaluation, which are presented in Section 4.8, show that discourse structure is useful for predicting coreference candidates that are likely to participate in intraprocedure coreference relations in instructions steps at the same level. Genre features can be used to effectively rank coreference candidates in installation procedures.

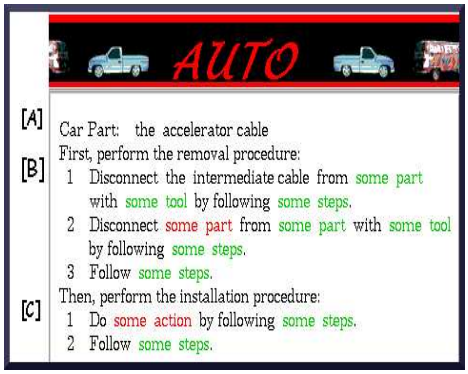


Figure 4.1: Partially Specified Car Repair Procedure.

4.2 Policy Type for Identifying Coreference Relations and Candidates

A coreference mechanism may use a **proactive** or **reactive** policy type to identify potential coreference relations and the candidates that may take part in these relations. A proactive policy type uses general types, the anchors in the feedback text that represent the top of the ontology hierarchy. *Some part* is an example of a general type. A reactive policy type uses specific values of general types, the leaves of the ontology hierarchy. For example, *intermediate cable*, is a specific value for the general type *some part*. Prior work (van Deemter & Power 1998) has considered a proactive policy type. This section describes systems using both proactive and reactive policy types to identify coreference relations and candidates.

When a **proactive** policy type is used, the system anticipates coreference at anchors using the local context. All unspecified anchors that are of the same type as an anchor that has already been specified represent locations where coreference is possible. For example, in Figure 4.1, locations where an anchor of type *part* has yet

to be specified ([B1] and [B2]) are locations where it is possible to refer again to the accelerator cable ([A]) or the intermediate cable ([B1]), the *part* anchors that have already been specified.

When a **reactive** policy type is used, the system waits to identify potential coreference sites until experts specify the value of an anchor. For example, in Figure 4.1, the system waits until experts specify *intermediate cable* as the value of *some part* in [B1] or [B2] before identifying that the value of one of these arguments may corefer to the intermediate cable already specified in [B1]. If a reactive policy type is used, the system identifies fewer potential coreference sites and fewer coreference candidates at each site.

4.3 Determining Relative Saliency of Coreference Candidates

Once the candidates that may take part in coreference relations at potential coreference sites have been identified, they must be organized and then presented to experts. The coreference mechanisms first determine those objects that are salient at the point in the procedure where experts wish to specify the value of an anchor. To compute saliency, the coreference mechanisms take into account two factors: discourse structure and genre features. This information is then used to determine the objects that are most likely to be referred to again.

4.3.1 Factor 1: Discourse Structure

Discourse structure is one of the factors that the coreference mechanisms may bring to bear to determine the saliency of entities. This section provides background on theory of discourse structure (Grosz & Sidner 1986) that the coreference mechanisms may employ, and describes how AUTO uses this theory to rank coreference candidates at potential coreference sites. The Grosz and Sidner theory of discourse structure is the foundation for representing discourse structure as a hierarchy of segments. In this theory, discourse structure consists of three interrelated components: 1) *linguistic structure*, 2) *intentional structure*, and 3) *attentional state*. The *linguistic structure* of a discourse comprises the discourse segments themselves and the hierarchical relationships that may hold between them. Each discourse segment consists of one or more locally coherent utterances of the discourse. Linguistic devices are often used to mark discourse segment boundaries. Previous work has shown that discourses in many domains, among them task-oriented dialogues (Grosz 1978),[3] can be decomposed into discourse segments. The *intentional structure* of a discourse reflects the purpose of the discourse segments and determines hierarchical relationships among them. The relationship that holds between elements of the intentional structure is outwardly reflected by the linguistic structure. The discourse has an overall discourse purpose; and each discourse segment has a discourse segment purpose, which contributes to the discourse purpose. A dominance relationship holds between discourse segment purposes when the satisfaction of one discourse segment purpose contributes to another. The *attentional state* represents the focus of attention, formalizes the no-

[3] A task-oriented discourse describes a physical task to be performed.

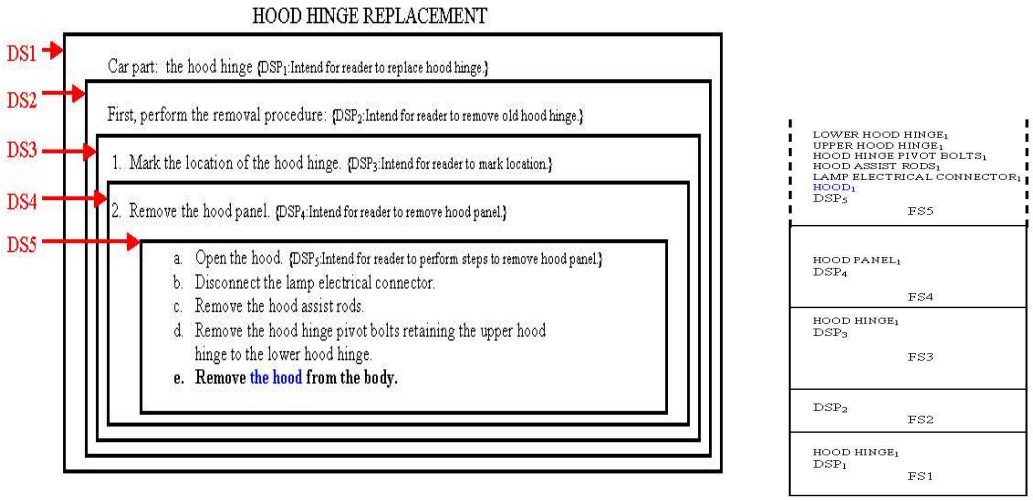


Figure 4.2: Use of Discourse Structure to Predict Coreference: Linguistic and Intentional Structure (a) and Focus-Space Stack Just Before the Specification of Step 2e of the Removal Procedure (b).

tion of object salience, and explains how context affects the processing of utterances in the discourse. At the local level, the attentional state is modeled by centering theory (Grosz, Joshi, & Weinstein 1995), which addresses how intra-segment utterances are related. The global level of the attentional state is represented by a stack of focus spaces (Grosz 1977),[4] which models the relations between discourse segments. There is a one-to-one mapping between discourse segments and focus spaces. A focus space contains the entities that are salient in the discourse segment corresponding to the focus space and a representation of the discourse segment purpose. Pushes and pops of focus spaces follow the hierarchical structure of the discourse segments.

Discourse theory makes several interesting predictions regarding the saliency of ob-

[4] Proposed alternatives to representing the attentional state by a stack mechanism include the cache model (Walker 2000; 1996) and Veins Theory (Cristea, Ide, & Romary 1998; Ide & Cristea 2000).

jects and coreference. As each focus space representing a discourse segment is pushed onto the focus-space stack, those objects referred to in the current focus space are most salient. Objects mentioned in other focus spaces lower in the stack are also accessible, but less salient. In terms of the linguistic structure, an anaphor's antecedent is more likely to have been mentioned hierarchically recently in the discourse (Cristea *et al.* 1999). The antecedent of a pronoun, for example, is most likely to be found locally, within the segment the pronoun is used. The utterances in this segment are about the same topic, i.e., they share the same discourse segment purpose. If an already mentioned entity is referred to using a definite description, its antecedent is somewhat more likely to be found in the current segment, but it may also be found in the discourse segments that hierarchically precede the current segment.

AUTO's coreference mechanisms exploit the current discourse structure to inform the algorithm used to rank coreference candidates. The manual text that domain experts interact with replaces the human discourse analyzed by discourse structure theory. The mechanisms retain a discourse model that contains all of the entities up to the point at which experts are currently choosing an object. If this location is deemed to be a potential coreference site, the position of entities in the discourse model, i.e., their saliency, is used to rank possible antecedents. Figure 4.2(a) shows a partially specified repair procedure in the task-oriented domain of car repair procedures. Discourse segments [the linguistic structure] have been delineated and discourse segment purposes (DSPs) [the intentional structure] have been labeled. Figure 4.2(b) shows the focus space stack just before the specification of Step 2e of the removal procedure. The stack contains focus spaces corresponding to discourse segments 1, 2, 3,

4, and 5 (DS1, DS2, DS3, DS4, and DS5) because in the dominance hierarchy, DS1 *dominates* DS2, DS2 *dominates* DS3, DS3 *dominates* DS4, and DS4 *dominates* DS5. In specifying Step 2e of the removal procedure, the domain expert is about to refer again to the *hood* that was initially specified in Step 2a. This object is predicted to be quite salient, since it is present in the current focus space, FS5.

Domain experts are more likely to refer repeatedly to salient entities contained in instruction steps about the same topic as a given instruction step. For this reason, AUTO ranks coreference candidates based on their saliency, as dictated by the discourse structure of the current manual and the current focus-space stack, taking into consideration the peculiarities of object specification order, before presenting linguistic expressions referring to them in menus. Discourse structure theory is used to impose preferences on potential antecedents, but because any entity may be referred to again, the menus contain all entities of the appropriate type that have already been specified. Unlike speech and English text, in which information is processed form left to right, in knowledge editing, entities can be introduced in any order. Therefore, menus for anchors in the beginning of a document may contain linguistic expressions for entities that have already been specified later in the document.[5]

Figure 4.3 shows the hierarchical discourse structure of a partially specified car repair procedure. For each potential coreference relation that AUTO identifies, discourse structure is used to rank coreference candidates. Entities of the *same* type appearing in the same discourse segment are deemed most likely to be referred to

[5]Though knowledge-editing systems allow for knowledge to be specified in any order, when using a WYSIWYM-based system such as AUTO, it is most common and natural to specify knowledge following a left to right order. This may be the case because experts interact with natural-language text to build the KB. In this respect, the knowledge-editing task is akin to the process of writing a natural-language document.

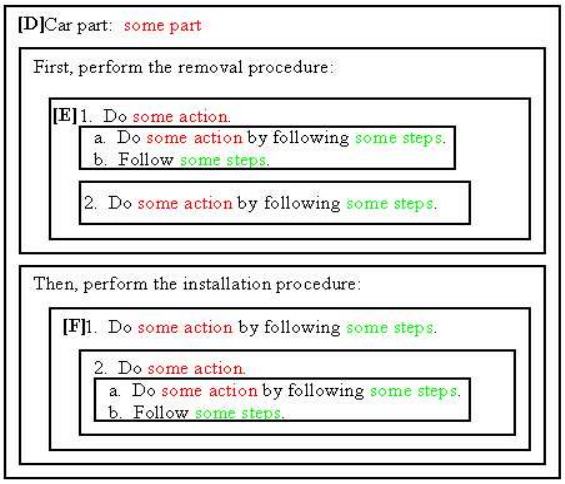


Figure 4.3: Car Repair Procedure with Discourse Segments Delineated.

again. For example, if an entity of type θ is specified in both [E1a] and [E2], the system predicts that if [E1b] contains another reference to an argument of type θ, experts will most likely refer to the entity mentioned in the current segment, [E1a], even though, in terms of linear proximity, both [E1a] and [E2] are equidistant from [E1b]. We measure linear proximity according to the number of noun phrases between the argument currently being specified and the potential coreference candidate. It is uncommon to make an initial reference to an object in an embedded segment and then to refer again to this object in a higher-level segment that follows. For this reason, following the current segment, the parent segment and then segments that hierarchically precede the parent segment are examined until reaching the top level of the hierarchy. Entities in the parent segment and the segments that hierarchically precede it, along with those entities in the current segment, are the entities that are currently contained in the focus-space stack. When entities of the same type in the parent segment and those that hierarchically precede it have been exhausted, the

algorithm looks to daughter segments, beginning with those that are embedded one level, then moving on to those that are embedded further in order of depth. The rationale for considering objects in parent segments to be more salient than those in daughter segments is that naturally, domain experts tend to specify car repair procedures in the same manner in which they write them in natural language–from left to right. It is unnatural for experts to specify steps in an embedded subprocedure before specifying the steps in the outer-level procedure that precede the embedded subprocedure. All remaining objects are ranked according to linear proximity, breaking ties by favoring objects specified earlier in the document than the current instance.

4.3.2 Factor 2: Genre Features

Discourse structure theory is a good predictor for coreference when an entity is mentioned hierarchically recently. Sometimes, though, the ranking of coreference candidates that the algorithm imposes based on the current discourse structure is inaccurate. Figure 4.4(b) shows the focus-space stack just before Step 2 of the installation procedure in Figure 4.4(a) has been specified. In Step 2, the domain expert wishes to refer again to the hood that was mentioned in Steps 2a, 2d, and 3 of the removal procedure. In this example, discourse structure theory predicts that the hood object is not salient at this point in the procedure. The focus spaces in which the previous references to this object are contained, FS5 and FS6, have already been popped from the stack. To account for interprocedure coreference, the mechanisms take a second informational factor, genre features of the task-specific car repair domain, into account when organizing coreference candidates. The influence of the nature of

the task on the structure of car repair procedures became evident after examining the procedures randomly chosen for the user study presented in Chapter 3. Of the 44 unique car part types contained in the removal procedures of the eight car repair procedures, 39 (almost 89%) of these part types were repeated in the installation procedure of the same replacement procedure. Over 75% of these cases where repeated mentions of the same instance, i.e., cases of coreference. Figure 4.5 shows an example of the multiple coreference relations spanning the removal and installation procedures of the replacement procedure for the air inlet grille panel. Brackets and subscripts have been added to highlight coreference relations. To account for the influence of task structure on coreference, the system constructs a *Removal Procedure Parts* menu to present to experts when coreference is possible in the installation procedure. This menu contains a list of the parts that were referred to in the removal procedure of the current replacement procedure.

The part referred to in the correct argument position of the removal procedure instruction step that most likely corresponds to the inverse of the installation procedure instruction step currently being specified is listed first. The most likely inverse of the current step is determined by assuming that the installation procedure will be the exact inverse of the specified removal procedure. More specifically, each instruction step of the installation procedure is predicted to be the inverse of its corresponding step in the removal procedure. These inverse installation procedure steps are expected to appear in the opposite order of their corresponding steps in the removal procedure. For example, in Figure 4.4(a), the **penultimate** step in the removal procedure is *Remove the bolts retaining the hood hinge to the hood.* The inverse of this step is pre-

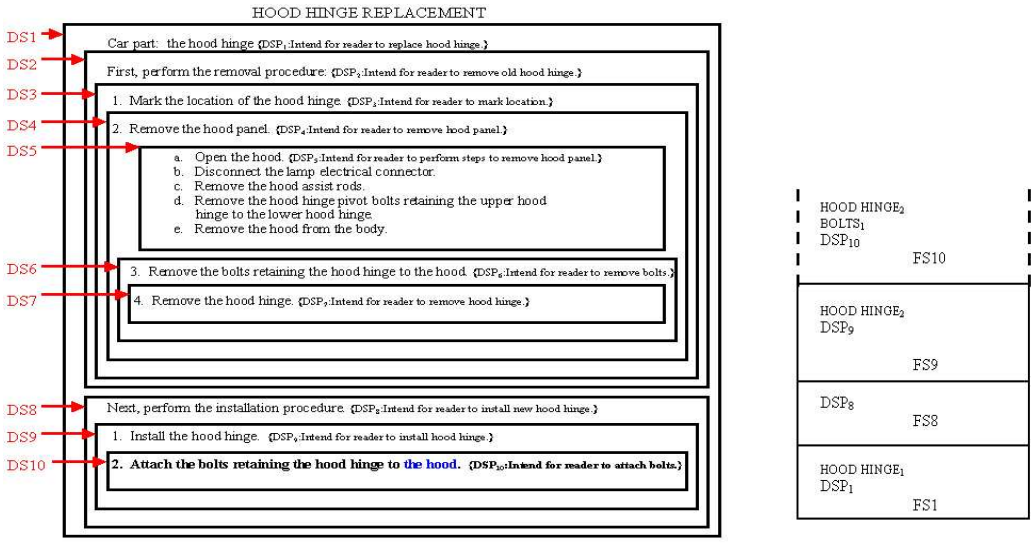
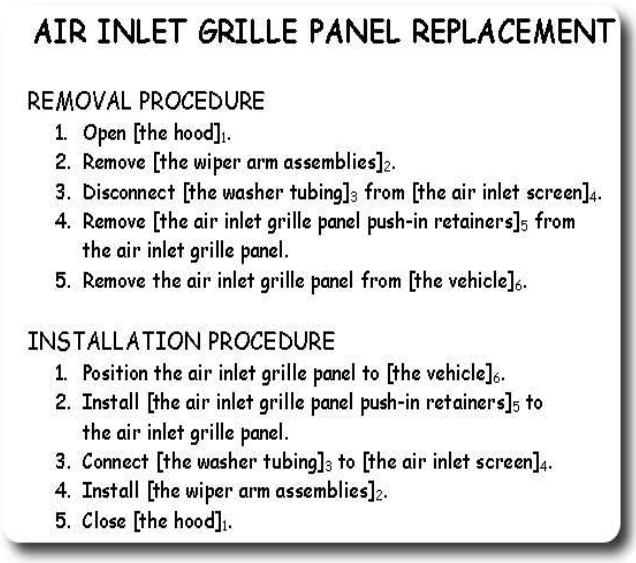


Figure 4.4: Use of Discourse Structure to Predict Coreference: Linguistic and Intentional Structure (a) and Focus-Space Stack Just Before the Specification of Step 2 of the Installation Procedure (b).

dicted to appear as the **second** step in the installation procedure. The entity most likely to be referred to in the second step of the installation procedure, therefore, is the set of bolts that retain the hood hinge to the hood. Two caveats must be taken into account. First, if the entity that is likely to be referred to in an instruction step is an entity of the same type as the part being replaced, this is not likely to be an instance of coreference. Instead of being reused like most of the other parts, the part being replaced is discarded and replaced with a new one of the same type. Second, argument order within instructions steps must be preserved. For example, the inverse of Step 2.e in Figure 4.4(a) is likely to be *Install the hood to the body.* The specification order of the two part arguments in the inverse step predicted to appear in the installation procedure is the same as their order in the corresponding removal



AIR INLET GRILLE PANEL REPLACEMENT

REMOVAL PROCEDURE

1. Open [the hood]$_1$.
2. Remove [the wiper arm assemblies]$_2$.
3. Disconnect [the washer tubing]$_3$ from [the air inlet screen]$_4$.
4. Remove [the air inlet grille panel push-in retainers]$_5$ from the air inlet grille panel.
5. Remove the air inlet grille panel from [the vehicle]$_6$.

INSTALLATION PROCEDURE

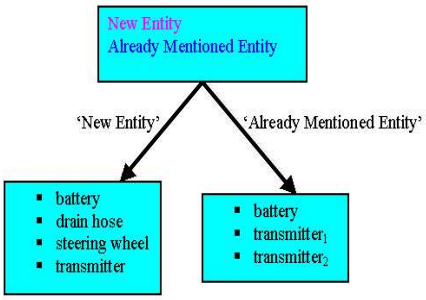
1. Position the air inlet grille panel to [the vehicle]$_6$.
2. Install [the air inlet grille panel push-in retainers]$_5$ to the air inlet grille panel.
3. Connect [the washer tubing]$_3$ to [the air inlet screen]$_4$.
4. Install [the wiper arm assemblies]$_2$.
5. Close [the hood]$_1$.

Figure 4.5: Coreference Relations Spanning Removal and Installation Procedures.

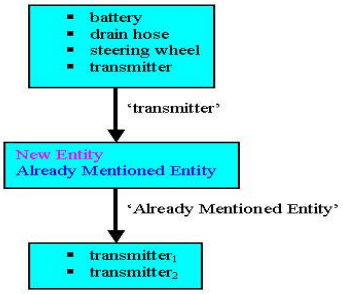
procedure step. Third, the higher-level step immediately preceding an embedded subprocedure is used to introduce the subprocedure. This higher-level step introduces the embedded subprocedure in the removal procedure as well as the inverse of the embedded subprocedure in the installation procedure. For instance, in Figure 4.4, Step 2 of the removal procedure introduces the embedded subprocedure for removing the hood panel. The inverse of this step is predicted to introduce the inverse of this subprocedure, namely a set of instructions for installing the hood panel, in the installation procedure. All other parts specified in the removal procedure are organized by proximity, in number of noun phrases, to the appropriate argument in the instruction step in the removal procedure predicted to be the inverse of the step currently being specified.

4.4 Communicating with Domain Experts to Determine Coreference Relations

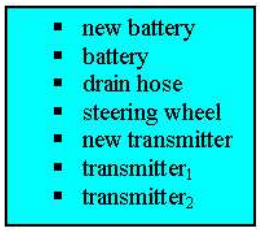
After the potential coreference sites have been located and the coreference candidates have been identified and organized, the user interface must facilitate the communication of experts and the system coming to agree on coreference relations. This section investigates two interaction styles: *direct manipulation* and *dialogue oriented*. With direct-manipulation interaction (Bouayad-Agha *et al.* 2002; Hana 2001), each linguistic expression that refers to an entity has a copy operation associated with it. If experts copy and then paste the expression in a location where an entity of the same general type is currently unspecified, the system assumes that experts wish to refer to the same entity again. The system adheres to the experts' commands, extending the KB and generating text to represent the specified coreference relation. With dialogue-oriented interaction, the system presents experts with menus from which selections are made. The dialogue interaction style can deploy two methods for presenting coreference candidates: *status first* and *status concurrently*. When status first is used, experts specify whether the object is new or has already mentioned before examining coreference candidates. Status concurrently, however, displays menus containing both new and already mentioned objects together. Each method can be used with either a proactive or reactive policy type. When a reactive policy type is used, menus containing coreference candidates contain only specified objects of the same part type, such as all of the specified *accelerator cables*. The use of a proactive policy type results in menus containing coreference candidates for anchors of the



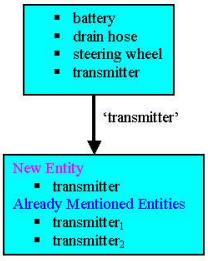
(a) Status First–Proactive



(b) Status First–Reactive



(c) Status Concurrently–Proactive



(d) Status Concurrently–Reactive

Figure 4.6: Menus for Specifying *Some Part* Using Different Policy Types.

same type that have currently been specified, such as all of the *part* objects that have been specified. Figure 4.6 shows the menus that result when the different dialogue interaction methods are employed. For this example, the *part* objects contained in the system's ontology include the following: battery, drain hose, steering wheel, and transmitter. The KB contains three objects: two of type *transmitter* and one of type *battery*.

When **status first–proactive policy type** is used (Figure 4.6(a)), experts are first asked if the *part* anchor being specified is a new or already mentioned entity. If they choose *New Entity*, they then choose from a list of all parts contained in the ontology. After choosing a part from this menu, a KB object of the chosen type is

created. If they choose *Already Mentioned Entity*, next, they choose from a list of referring expressions for each KB object currently in the KB. When **status first–reactive policy type** is used (Figure 4.6(b)), experts first choose a part from a list of all parts contained in the ontology. Then, they are asked if the entity is new or already mentioned. A *New Entity* selection results in a KB object of the chosen type being created. If they choose *Already Mentioned Entity*, they then must choose from a list of instances of the type that they chose.

When **status concurrently–proactive policy type** is used (Figure 4.6(c)), experts choose from a single menu containing all new and already mentioned parts. When **status concurrently–reactive policy type** is used (Figure 4.6(d)), experts first select a part from a list of all parts contained in the ontology. Then, they choose from a menu containing both new and already mentioned parts of this type.

Direct manipulation and status-first dialogue interaction with reactive policy type have been used in previous WYSIWYM-based systems (van Deemter & Power 1998), while status concurrently is a new method. There are several trade-offs involved in using the four dialogue-oriented methods. Allowing a user to reach desired selections quickly has been shown to be a desirable property for a user interface (Shneiderman 1992; Johnson 2000). Of the dialogue-oriented methods, status concurrently–proactive policy type allows experts to specify selections in the fewest number of system queries. As more KB objects of the same type are created, however, menus have the potential to become quite long; therefore, methods employing status first may be preferable. From the system's perspective, a status first–reactive policy type minimizes, on average, the amount of information that has to be sent to the menu

construction module. If experts wish to refer to an already mentioned entity, only information regarding the currently specified parts of the type they desire to refer to again need to be sent to the menu construction module. Status concurrently–proactive policy type, on the other hand, requires that unnecessary information be sent to the menu construction module. For example, information regarding all of the new and already mentioned entities is sent to the menu construction module even though, for a given anchor, only information regarding either new or already mentioned entities is necessary.

4.5 Representing Coreference Relations in the User Interface

The system must effectively distinguish new entities from existing ones in text presented to domain experts. Some systems for writing natural-language specifications in a controlled manner borrow constructs from programming languages to display coreference relations in user interfaces (Macias & Pulman 1995). More specifically, LET statements associate a global name with an entity, and, throughout the text, the system uses this name in all locations where the entity is mentioned. Previous WYSIWYM-based systems have used centering theory (Grosz, Joshi, & Weinstein 1995) and a unification grammar for generating pronouns and noun phrases with ordinals, such as *the first document* and *the second document*, in the feedback text to clearly denote objects (Power 1999; Bouayad-Agha *et al.* 2002; Kibble & Power 2000). Previous work has also investigated the use of artificial ele-

ments such as brackets with indices for clearly presenting coreference relations (van Deemter & Power 1998). In AUTO, representing coreference is important in the menus presented to experts, both in the menu of all entities that have been mentioned and in the shorter menu of parts referred to in the removal procedure, and the feedback text describing the content of the KB. The menus and feedback text may use different elements for representing coreference. This section investigates the use of two types of elements for representing coreference in the user interface: *linguistic* and *artificial*.

The linguistic elements are full noun phrases and reduced noun phrases. If more than one instance of the same type exists, ordinal numbers are used in the feedback text and menus. For example, if a manual contains two instances of type *tie wrap*, the system uses *the tie wrap* to refer to the first one (in terms of specification order) and *the second tie wrap* to refer to the second one in the feedback text and menus. When an instance of the same type already exists, the system uses the difference words *other* or *another* to refer to *new instances*. For instance, the system uses the expression *another tie wrap* for a menu selection representing a new tie wrap instance. When reduced noun phrases are used, the referring-expression generation algorithm described in Chapter 5 for generating referring expressions in the output text is used to generate reduced noun phrases in the feedback text and menus. Reduced expressions are used to signify the link to previous information contained in the manual. Domain experts are likely to prefer reduced forms since the full noun phrase is often not the noun phrase that they look for in scanning a menu for a desired selection. If more than one instance of the same type exists, ordinal numbers are used in the same way

they are used with full noun phrases. Another linguistic element that could be used in menus is relative clauses. Menus would need to contain identifying phrases that clearly distinguish each option from all others. Though relative clauses are harder to generate, an advantage to using them is that experts would not have to match the object in the menu with its mention in the text to determine which object is being referred to in the menu. Instead, the relative clause would provide context for determining the referent of the expression. For example, the menu item *the park lock cable that was removed* clearly indicates which park lock cable is being referred to whereas the menu item *the park lock cable* forces experts to consult the text to determine if the expression refers to the park lock cable that they specified should be removed or the one that they specified should be installed. Though linguistic elements may lead to more coherent and natural-sounding text, they may introduce ambiguity and are not as precise as artificial elements. Because the feedback text must be clear and unambiguous to ensure that the correct output text is generated, the coreference mechanisms may also use artificial elements. The artificial elements that the mechanisms may use to represent coreference relations are brackets with indices.[6] When brackets with indices are used, noun phrases in the text with the same index corefer. For example, all occurrences of $[\text{bolts}]_1$ in the text refer to the same entity.

[6]Color is another artificial element that could be used. In the Alembic Workbench (Day *et al.* 1997), a graphical user interface for annotating texts, annotated coreference chains are highlighted using different colors.

4.6 Systems that Rank Previously Mentioned Options

The properties of **policy type**, **communication style**, and **UI content** are generally decided on by the system designer. The algorithm used to rank coreference candidates, however, can be empirically evaluated. To determine the contribution of discourse structure and genre features in informing the ranking of coreference candidates, we created three versions of AUTO extended to handle coreference. In organizing already mentioned options in menus presented to experts as they edit a KB, AUTO+disc takes advantage of discourse structure; AUTO+genre, genre features. To demonstrate the usefulness of the coreference candidate ranking algorithms employed by AUTO+disc and AUTO+genre, we compared them to a system that organizes coreference candidates alphabetically, the manner used in prior WYSIWYM-based systems that support coreference (van Deemter & Power 1998). The three systems employ coreference mechanisms with common property values presented in Table 4.1. The dimension along which the versions differ is the method used to rank coreference candidates. An example menu produced by each version when *some part* in Step 2.2 of the water deflector installation procedure in Figure 4.7 is being specified is shown in Figure 4.8. Step 2.2 must convey that the *trim panel retainers*, most recently mentioned in installation procedure Step 2.1, need to be tightened. A brief description of each system follows:

1. **AUTO** (Figure 4.8(a)), extended to support coreference, uses an uninformed method for organizing all of the part object coreference candidates. The system

Property	*Value*
Policy Type	Proactive
Communication Style	Dialogue Oriented with Status First
UI Content: Menus	Full Noun Phrase
UI Content: Feedback Text	Full Noun Phrase

Table 4.1: Property Values for Coreference Mechanisms in Figures 4.7 and 4.8.

alphabetizes the candidates, ignoring ordinals, and presents them in the *Already Mentioned Parts* menu. Since there are nine unique car parts that have already been specified in the repair procedure in Figure 4.7, the menu presented in Figure 4.8(a) contains nine coreference candidates. The desired candidate, the trim panel retainers, is the sixth item in the list.

2. **AUTO+disc** (Figure 4.8(b)) examines the discourse structure to organize coreference candidates. The *Already Mentioned Parts* menu, like the one presented by AUTO, contains all of the nine unique car parts that have already been specified, but it is organized using the discourse structure ranking algorithm. The part entities in the current segment, the embedded subprocedure detailing how to install the trim panel to the door, are listed first. The only such entity in this segment is *the trim panel retainers.* Next, entities present in the parent segment and those segments that hierarchically precede it are listed. These parts include *the door* and *the trim panel* from the parent segment, *the second water deflector* from the segment that hierarchically precedes the parent segment, and *the water deflector* from the segment hierarchically preceding the grandparent segment of the current discourse segment. The rest of the

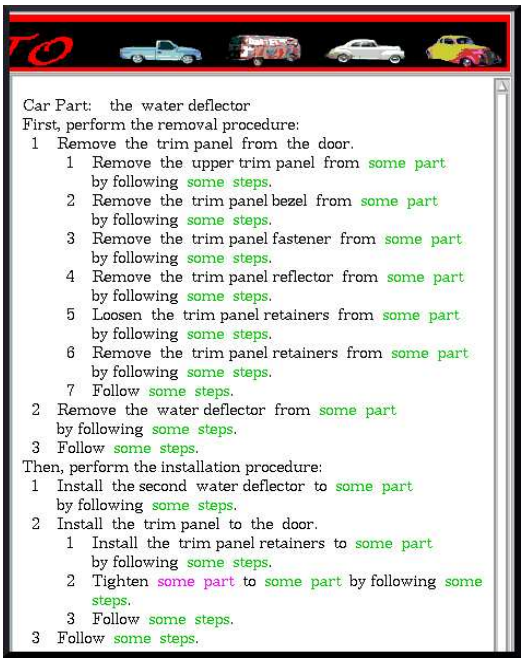


Figure 4.7: Interaction with Coreference Mechanism with Property Values in Table 4.1.

parts that have already been specified in the current procedure are organized according to linear proximity to the part argument currently being specified in installation procedure Step 2.2.

3. **AUTO+genre** (Figure 4.8(c)) uses genre features of the task-specific car repair domain to organize coreference candidates. At points where coreference is possible in the removal procedure, the system presents the *Already Mentioned Parts* menu. When coreference is possible in the installation procedure, the system presents two menus: the *Already Mentioned Parts* menu and the *Removal Procedure Parts* menu. The *Already Mentioned Parts* menu, like the one presented by AUTO, contains an alphabetized list of the car parts that have already been specified. The *Removal Procedure Parts* menu, the one shown in

(a) AUTO (b) AUTO+disc (c) AUTO+genre

Figure 4.8: Menu Organization for Specifying *Some Part* (Installation Procedure Step 2.2 in Figure 4.7).

the figure, contains the eight unique parts that were specified in the removal procedure. The *trim panel retainers* option is listed first. Step 2.2 is predicted to be the inverse of Step 1.5 of the removal procedure. The first part argument of that step, *trim panel retainers* is also predicted to be the first part argument of Step 2.2 in the installation procedure. The seven other parts are ranked according to their proximity in the removal procedure to the *trim panel retainers*.

4.7 Experimental Design

The main goal in evaluating AUTO extended to support the repeated mention of entities during KB specification is to determine if subjects find the organization of coreference candidates arrived at as a result of analyzing discourse structure and genre features useful and to characterize situations in which different coreference candidate ranking methods are preferred. We compared the three systems for authoring car repair procedures described in Section 4.6: AUTO, AUTO+disc, and

AUTO+genre. AUTO, like existing knowledge-editing systems that support coreference (van Deemter & Power 1998), organizes coreference candidates alphabetically. The other two systems use informed methods that exploit discourse structure and genre features. Comparing AUTO to AUTO+disc highlights the contribution of the discourse structure, whereas the comparison between AUTO and AUTO+genre makes the information provided by genre features evident.

The following three factors were controlled in the study:

1. **System**: All subjects interacted with three systems: AUTO, AUTO+disc, and AUTO+genre. None of the systems supported *copy* and *paste* editing operations. The systems did, however, have available *undo* and *cut* operations for correcting mistakes.

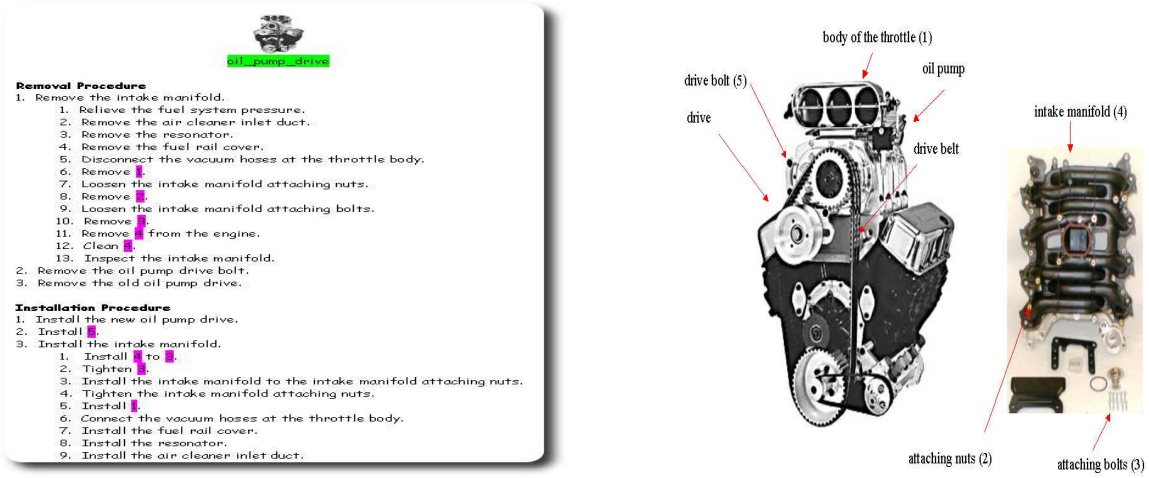
2. **Procedure**: From a held out set of procedures,[7] we chose three car repair procedures that highlight the contexts in which coreference is common in procedural domains. First, in instruction steps at the same level, anaphoric references most often refer to entities most recently mentioned within the previous one or two instruction steps. Second, when an expression corefers to an entity most recently mentioned in a higher-level procedure, the antecedent of the anaphoric reference is often most recently referred to within the previous one or two instruction steps of the higher-level procedure. Third, in the car repair domain, interprocedure coreference is common. The installation procedure of a car repair procedure is often the exact inverse of the removal procedure.

[7]None of the procedures were part of the collection used to determine categorical membership or to create the statistical model.

Each subject authored all three repair procedures, one per system. Each procedure contained removal and installation subprocedures. The structure of each procedure was nested, i.e., some of the instruction steps contained embedded procedures. All anchors marking where knowledge could be specified were *red*, however, subjects were told that they did not need to specify a value for all of the anchors. Subjects were provided with a repair procedure description and a diagram, examples of which are shown in Figure 4.9(a) and (b), respectively. In the descriptions, select car parts are replaced by numbers. The numbers correspond to the labeled parts in the diagram with these same numbers. Where the diagram context allows, reduced noun phrases are used as part labels in the diagram. This is done to mimic the cognitive task involved in authoring a car repair procedure, thus preventing subjects from knowing the exact noun phrase referring expression the system uses to name parts. Equipped with the noun phrase the system uses to name a part, the subject would simply need to find this exact noun phrase in the list of parts. Subjects were instructed to specify all parts represented by numbers in the description. All of the numbered parts in the description were cases of coreference to a part mentioned earlier in the procedure. Subjects were given the template in Figure 4.10 as a starting point for specifying the description in Figure 4.9(a).

3. **Order**: For the experiment, subjects were randomly divided into three groups of three subjects. A Latin Square[8] (Martin 2003) determined the order in which the different groups of subjects interacted with the systems. Within each group

[8] A balanced Latin Square does not exist for a 3x3 experimental design.



(a) (b)

Figure 4.9: Description (a) and Diagram (b) for the Oil Pump Drive.

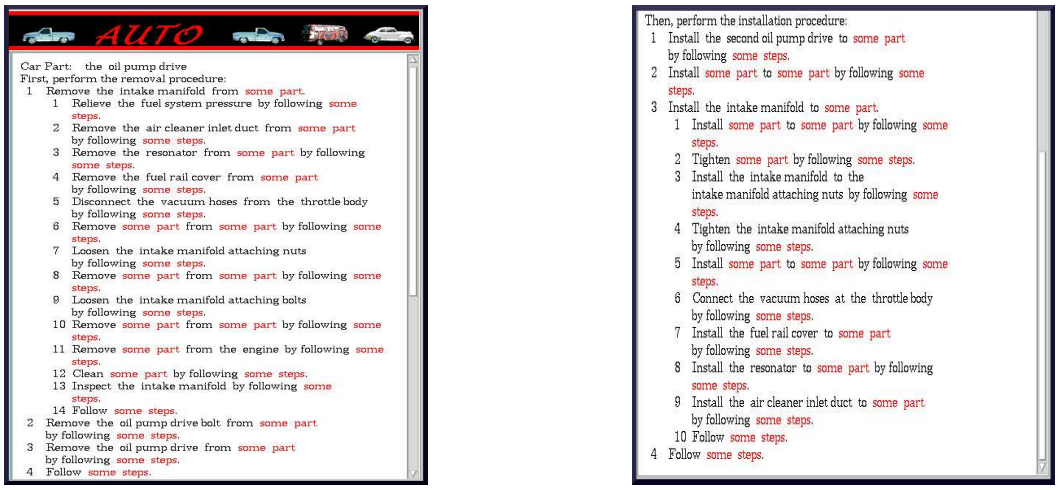
of subjects, a Latin Square also dictated the order in which subjects authored the three procedures.

Nine subjects[9] interacted with each system to author a car repair procedure. They filled out the short questionnaire in Appendix B to assess user satisfaction after interacting with *each* system. For each subject/system/procedure triplet, the system logged selection time,[10] the elapsed time between when the subject was presented with a list of options and when the subject made a selection, for each car part option. For AUTO and AUTO+disc, the system initially presented the contents of the *Already Mentioned Parts* menu; for AUTO+genre, the *Removal Procedure Parts* menu.[11] Only

[9]The subjects were graduate students who had neither participated in the user study described in Chapter 3 nor had experience using AUTO. They were, however, given a tutorial prior to using each system. The tutorial they were given before using AUTO+disc can be found in Appendix B.

[10]Subjects were not explicitly told they were being timed, but they were encouraged to "write procedures as fast as possible while maintaining accuracy." Also, they were made aware that they had a time limit for using each system to author the procedure they were given.

[11]Note that in Chapter 3, the system did not present a list of options initially because an aim of



Removal Installation

Figure 4.10: Template for the Oil Pump Drive Description and Diagram in Figure 4.9.

expressions referring to entities mentioned before the selected anchor were presented in the menu. The fact that entities following the selected anchor in the procedure have already been specified is an anomaly of the experimental design. It is most natural for subjects to specify procedures left to right, therefore, making the presence of selected entities following the selected entity unlikely to occur in practice. For each of the three systems, the experiment resulted in the authoring of nine procedures and the collection of 93 repeated car part selection times.

4.8 Results and Discussion

We determined the contribution of discourse structure and genre features by evaluating the average rank of the correct option in menus presented by the systems. We

the study was to determine if subjects could determine which menu contained the correct selection. Here, we are interested in subjects' ability to find the correct selection within a given menu.

examined option position within menus since, for the most part,[12] all systems present menus containing the same options. The results show that discourse structure is useful for predicting coreference candidates in removal procedure instruction steps at the same level. For coreference in the installation procedure, genre features can be used to effectively determine the most likely coreference candidates.

4.8.1 Contribution of Discourse Structure

Table 4.2 compares the average rank of correct options for all parts subjects were required to specify in the study. On average, subjects had to examine 56% of the menu before arriving at the correct option when using AUTO. When discourse structure was used to rank coreference candidates in AUTO+disc, the correct option could be found by examining only 47% of the menu, a reduction of 9% of the options examined using AUTO. Table 4.3 shows that the reduction in the percentage of options that had to be examined did not lead to a significant reduction in selection times. The reason for this is two-fold. First, the repeated mention menus presented to subjects in the study were fairly short, having an average length of 12.1 $\pm$ 4.0 options. A significant reduction is likely to be observed when authoring longer procedures, which have more coreference candidates associated with each part anchor. Second, Table 4.4 shows that discourse structure is more useful for coreference in the removal procedure. For removal procedure parts, subjects had to examine 41% of the menu options when discourse information was used, compared to 56% when coreference candidates were organized alphabetically. In the installation procedure, the first instance

[12]The only exception being *Removal Procedure Parts* menus, which may contain parts already mentioned in the installation procedure but not present in the removal procedure.

of coreference for each part is the result of the part having been mentioned in the removal procedure. The discourse segment in which the removal procedure instruction steps are contained has been popped off of the focus space stack. Therefore, the entities mentioned in the removal procedure are deemed not to be likely coreference candidates. Once a part has already been mentioned in the installation procedure, discourse structure is effective for predicting coreference candidates.

A more thorough analysis of instances of coreference in the removal procedure reveals that discourse structure information is most useful for predicting candidates for coreference relations in instruction steps at the same level. Figure 4.9(a) illustrates why this is the case. Removal procedure Steps 1.11-13 give more specific details about the information that was summarized in the instruction step introducing the embedded procedure, Step 1. In these steps, the entity that is the focus of the embedded procedure instruction step, the *intake manifold*, is referred to again. The menu option for this entity is ranked following all of the entities mentioned in the current embedded subprocedure. If only coreference relations in removal procedure instruction steps at the same level are taken into account, subjects had to examine a significantly ($p < 0.04$) smaller percentage of menu options when discourse structure was used (36.5% $\pm$ 25.5%), compared to when they were alphabetized (56.9% $\pm$ 26.1%).

4.8.2 Contribution of Genre Features

Table 4.5 presents the percentage of the repeated mention menu that had to be examined to locate the correct part for all specified parts in installation procedures

Procedure	*AUTO*	*AUTO+disc*
1	60.8% ± 17.7%	29.1% ± 29.7%
2	65.6% ± 28.3%	52.4% ± 30.0%
3	37.8% ± 23.8%	59.9% ± 28.0%
Average	56.0% ± 26.1%	47.1% ± 31.2%

Table 4.2: Comparison of Average Rank of Correct Option (% of Menu Size) for all Specified Parts: AUTO vs. AUTO+disc ($n = 31$).

Object Type	*AUTO*	*AUTO+disc*	*AUTO+genre*
car parts	23.7 ± 4.9	23.4 ± 4.5	22.1 ± 5.3

Table 4.3: Average Selection Time (in Seconds) for Car Parts Specified Using Three Versions of AUTO.

Procedure	*AUTO*	*AUTO+disc*
1	59.8% ± 16.3%	31.6% ± 38.4%
2	60.0% ± 28.5%	49.4% ± 28.1%
3	39.7% ± 27.5%	43.2% ± 25.4%
Average	56.1% ± 24.9%	40.5% ± 29.8%

Table 4.4: Comparison of Average Rank of Correct Option (% of Menu Size) for Removal Procedure Parts: AUTO vs. AUTO+disc ($n = 16$).

Procedure	*AUTO*	*AUTO+genre*
1	61.8% ± 30.0%	24.8% ± 15.8%
2	76.8% ± 27.9%	40.3% ± 21.2%
3	37.0% ± 24.6%	11.2% ± 5.2%
Average	55.9% ± 28.3%	23.5% ± 18.0%

Table 4.5: Comparison of Average Rank of Correct Option (% of Menu Size) for Installation Procedure Parts: AUTO vs. AUTO+genre ($n = 15$).

using AUTO and AUTO+genre. These are the cases for which AUTO+genre presents *Removal Procedure Parts* menus. The table shows that the use of genre features led to a significant reduction ($p \approx 0$) in the percentage of the menu that subjects had to examine when comparing AUTO (56%) to AUTO+genre (24%). Further, even though the reduction in selection times in Table 4.3 is not significant, subjects believed that AUTO+genre helped them to write car repair procedures more quickly and easily than the other systems.

The analysis of the use of discourse structure and genre features for organizing already mentioned options reveals that a system that uses either discourse structure or genre features to predict coreference candidates generates menus in which the correct option is ranked higher, on average, than the menus produced by a system that uses alphabetical organization, the uninformed method used by traditional knowledge-editing systems. Discourse structure is more useful for predicting candidates for intraprocedure coreference relations in instruction steps at the same level. Genre features are effective for predicting coreference in installation procedure instruction steps.

4.9 Possible Extensions

Opportunities for extending the work presented in this chapter lie in the areas of coreference prediction, ranking coreference candidates, representing coreference relations, and refining the interpretation of editing operations in user interfaces that support coreference. Equipping AUTO with the capability to reuse procedures presents interesting challenges for coreference prediction. When a previously authored procedure is inserted within a procedure currently being specified, if an entity of the same type is referred to in both the previously generated procedure and the procedure currently being specified, the system must determine if these entities are the same instances or different instances of the same type to extend the KB correctly.[13] The system could distinctly mark chains of referring expressions likely to refer to the same entity based on rules acquired automatically and manually encoded heuristics. An example of a heuristic for determining whether or not referring expressions are coreferential is as follows: in a replacement procedure for a car part, if a part that is the same type as the part being replaced is mentioned in both the removal and installation procedures, these parts are unlikely to refer to the same instance. Instead, they are likely to refer to different instances of the same type–the one in the removal procedure being the malfunctioning part and the one in the installation procedure being the replacement part. For instance, if domain experts are authoring a replacement procedure for a tire, and after authoring the removal procedure, they insert a previously written installation procedure, both of which refer to a *tire*, these tires are unlikely to be the same. Domain experts must verify the correctness of the system's

[13]The research area of KB arbitration or merging addresses issues like this one (Liberatore & Schaerf 2000; 1998, inter alia).

coreference predictions before the output text for the manual can be generated.

This chapter investigated the use of two factors for ranking coreference candidates: discourse structure and genre features. It is worth addressing several other factors as well as the combination of these factors to accurately predict those entities that are most likely to be referred to again. It would be interesting to examine the use of other theories of discourse structure (Cristea, Ide, & Romary 1998; Ide & Cristea 2000, inter alia) and the proximity of candidates in terms of linear recency, the ordering policy often used to determine potential antecedents in anaphora resolution systems (Lappin & Leass 1994, inter alia), for ranking coreference candidates. Alternatively, the system could keep track of the order in which KB objects are created and then create a list of objects ordered based on closeness, in terms of specification time as opposed to hierarchical distance, to the argument currently being specified. Another factor that could be considered is closeness in physical proximity of objects inside a car, since mentioning objects brings other objects that are physically close to the mentioned object into focus, even though these objects may not be mentioned explicitly (Grosz 1977). Domain and lexical knowledge could also be considered. The statistical model developed in Chapter 3 could be used to further inform a model for predicting the most likely object to be referred to again. The statistical model takes advantage of knowledge at the sentence level by examining the selectional restrictions imposed by the context.

After developing models that take into account the aforementioned factors, the ranking of coreference candidates produced by the individual models could be combined into a mixed model.[14] The mixed model could use a metric that takes into

[14] Mixed models have been proposed in the literature for the task of anaphora resolution (Lappin

account the score that each coreference candidate receives from the evaluation metric of each individual model. The ranking produced by the mixed model provides an alternative to presenting the predications of the individual models in separate menus. In creating the mixed model, it would be interesting to evaluate the contribution of the different factors in predicting the most likely coreference candidates. Equipped with an accurate predication of which entity experts are most likely to refer to again, the system could take initiative and infer coreference relations automatically. If there are several candidates that are likely to participate in the coreference relation, the system could engage in a dialogue with experts until the system collects enough evidence to determine which entity experts wish to refer to again. Further, the model could be incorporated in anaphora resolution systems or other systems that require the identification of coreferential relations.

More accurately predicting the inverse of instruction steps from the installation procedure would lead to a more accurate ranking of coreference candidates in the *Removal Procedure Parts* menu. Currently, the system assumes that the installation procedure of a replacement procedure will be the exact inverse of the specified removal procedure to predict the instruction step from the removal procedure that is most likely to be the inverse of the current instruction step in the installation procedure. It would be interesting to use a more informed method for identifying the instruction step from the removal procedure that is likely to be the inverse of the current step. Relevant factors that could be taken into account include the verb specified in the current step and the steps already specified in the installation procedure, the surrounding context for the current step. The inverse step in the removal procedure

& Leass 1994; Rich & LuperFoy 1988; Carbonell & Brown 1988; Asher & Hajime 1988, inter alia).

is likely to contain a verb that is the opposite of the currently specified verb. For example, if the currently specified verb is *connect*, the inverse of this step is likely to contain the verb *disconnect*. The surrounding context indicates those actions which have already been performed. The predicted inverses of these steps can be eliminated from consideration for the inverse of the current step.

The ability to collapse embedded subprocedures to get a high-level view of the repair procedure presents challenges for representing coreference relations in the user interface. The system must be sure that the linguistic or artificial elements used to represent coreference relations are not dependent on material that is hidden as a result of collapsing the subprocedure. If so, the system must rerun the appropriate algorithms, taking only the currently visible text into account.

Cases of coreference in installation procedures are often unclear. In addition to the car part being replaced, it is often hard to distinguish other parts that are being discarded, not instances of coreference, from parts that are being reused, cases of coreference. It is important to clearly represent coreference relations in these cases because if an instance in which the referring expression is meant to corefer to an entity in the removal procedure is mistaken for an instance of coreference, the newly replaced car part may not function correctly.

This chapter focused on the editing operation of insertion. The system presents other editing options in terms of insertion. To correct mistakes, experts simply select the text that they want to revise. The system then presents a menu containing an insertion operation that results in the selected text being replaced by a phrase representing the entity one level higher in the hierarchy. For example, to replace an

incorrectly specified *negative battery cable* part with *a positive battery cable*, experts first select the phrase *negative battery cable*. Then, they choose the phrase *some part*,[15] the category one level higher in the hierarchy of parts, from the *General Part* insertion menu. This results in the text being regenerated to the state before the *negative battery cable* part was selected. Experts then select the *some part* anchor in the newly generated text. This selection results in a menu of parts, from which they can select the *positive battery cable*, the part they initially intended to specify. Prior work has provided a more thorough investigation of other editing operations with respect to coreference. Various user interfaces that support coreference have multiple versions of editing operations (van Deemter & Power 1998; Palm 1996, iter alia). In the Palm Desktop date book (Palm 1996), for example, when the user attempts to cut a repeating event, the system gives the user two options: to cut the current repeating event or all occurrences of the repeating event.

[15]This is assuming that they are using a version of AUTO that does not employ the car part categories described in Chapter 3.

Chapter 5

Reference in Document Generation

5.1 Introduction

Chapters 3 and 4 examined methods that allow domain experts to accurately refer to entities when specifying a KB using the WYSIWYM symbolic authoring approach. Once the KB has been completely specified, the system must define its content in a coherent natural language document, known as the output text. To generate coherent, natural-sounding text that allows the reader to correctly identify the entities referred to in it, the system must generate appropriate referential forms. Given the current document context, the referring expressions must unambiguously identify the intended entities for the reader.

This chapter describes models developed using a transformation-based machine learning algorithm for choosing referring expressions to be contained in the final document. It also presents an evaluation of the learned models. The parameter setting used to develop the top-performing learned model on the validation sets was used to

induce the final learned model for generating referring expressions. The performance of the final learned model on the held-out test set is compared to the performance of a baseline system, which always generates the full noun phrase referring expression. When compared to the baseline system, the learned model, which reduces expressions based on the surrounding context, predicts referring expressions that are more accurate and closer in length to those that people use.

Section 5.2 defines the problem of referring-expression generation, describes prior work in this area, and provides psycholinguistic evidence motivating the need to generate appropriate referential forms. Section 5.3 provides an introduction to transformation-based learning, the machine learning algorithm used to induce models for referring-expression generation. Section 5.4 describes the procedure used to obtain the gold-standard referring expressions used when framing the problem of referring-expression generation within the transformation-based learning paradigm, which is presented in Section 5.5. Section 5.6 presents the procedure used to automatically learn models for generating referring expressions. Section 5.7 analyzes the performance of the learned models on the five validation sets. In Section 5.8 the optimal parameter setting is used to develop the final model, which is tested on the unseen instances in the test set. Section 5.9 analyzes the predictions of the final model.

5.2 Problem Description: Referring-Expression Generation

The subproblem of referring-expression generation that this chapter examines is the generation of definite noun phrases. The induced models for referring expression generation, which are based on decisions made by people, are used to determine the content of expressions used to refer to entities in a specified KB. Based on the current context, the models determine whether or not a given word is to be included in a noun phrase. When choosing to reduce the number of words to include in a noun phrase, the models produce a noun phrase that signifies a link with a previously mentioned entity. The models generate expressions for performing initial and subsequent reference. Referring expressions generated by the models can be used to create cohesive, natural-sounding documents.

Prior work in the area of referring-expression generation has, for the most part, focused on either referring-expression form, in the case when a pronoun may be appropriate, or content, in the case of non-pronominal reference. Work that examines referring-expression form (Callaway & Lester 2002; McCoy & Strube 1999; Kibble & Power 2000, inter alia), which is often based on insights from centering theory (Grosz, Joshi, & Weinstein 1995), determines whether a pronoun may be used to refer to an entity that has already been mentioned. Work that address the content of referring expressions often uses an incremental algorithm (Dale & Reiter 1995) to build an expression to be used to perform initial reference. To construct an expression to refer to a given entity, the incremental algorithm searches for the set of attributes

that distinguishes the given entity from all other entities with which it might possibly be confused, the contrast set. The appropriate values for these attributes are then used to build a referring expression for the entity. Machine learning has been employed to learn which attributes to use for constructing these types of referring expressions (Jordan & Walker 2000). The incremental algorithm is based on the assumption that every entity can be characterized in terms of a collection of attributes and their values. For example, a small black dog can be characterized by the following attribute-value pairs: (size:small) and (color:black). In the car repair domain, it is not always possible to characterize entities, such as *air inlet grille panel push-in retainers*, in terms of attribute-value pairs. Further, the incremental algorithm focuses on creating referring expressions for entities that need to be distinguished from entities of the same type, such as creating an expression to refer to a small black dog that needs to be distinguished from a big brown dog and a small white dog. The algorithm generates expressions to identify whole objects. Though this type of reference must be performed in the car repair domain, more common is the need to generate expressions to refer to objects that are related in a different way: parts or components of a whole object mentioned within the context of the object. For example, in a context in which an air inlet grille panel has already been mentioned, the referring expression *push-in retainers* might be appropriate for referring to *air inlet grille panel* push-in retainers, a component of the air inlet grille panel.

The necessity that documents not contain unnecessarily complex referring expressions is supported by psycholinguistic studies (Gordon, Grosz, & Gilliom 1993; Gordon & Hendrick 1998, inter alia). When subsequent mentions simply repeat the

same referring expression, a "repeated-name penalty," leads to additional mental processing and, therefore, an increase in reading time. Expressions such as reduced noun phrases and pronouns provide a cue to coherence and lead to faster comprehension.

5.3 Background: The Transformation-Based Learning Paradigm

Transformation-based learning (Brill 1993) is the rule-based approach for automatically learning linguistic knowledge that this thesis applies to the problem of referring-expression generation. Transformation-based learning has been applied to many problems in the area of natural-language processing, including part of speech tagging (Brill 1995), prepositional phrase attachment ambiguity (Brill & Resnik 1994), and text chunking (Ramshaw & Marcus 1995). Figure 5.1 shows how a transformation-based machine learning algorithm works. Input to the algorithm includes 1) a training corpus of instances, 2) the correct class for each instance in the training corpus, and 3) a set of rule templates for creating possible rules. Learning a model using a transformation-based learning algorithm proceeds as follows:

1. Apply a baseline system to each instance in the training corpus to obtain the current corpus. The baseline system determines an initial hypothesis for the class of each instance in the training corpus. For part of speech tagging, an example initial guess is to assign each word the tag *noun*.

2. Use the current corpus, along with the correct class of the instances and the rule templates, to derive and score candidate rules. Rule templates include the

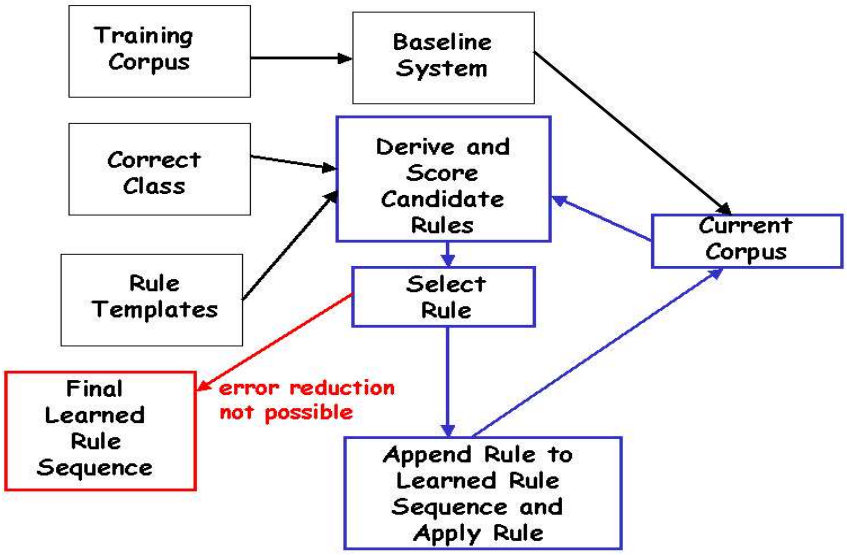


Figure 5.1: Transformation-Based Learning (Ramshaw & Marcus 1995).

combinations of features that can be used to form rules. They consist of two parts: a triggering context and a new class. To form a rule, the features need to be instantiated with values. For instance, an example rule template for part of speech tagging is: $POS\_TAG\_PREV => POS\_TAG\_CURRENT$ indicating that the triggering environment is the part of speech tag of the previous word, which determines the new part of speech tag for the current word. Instantiated with the values $POS\_TAG\_PREV = DT$ and $POS\_TAG\_CURRENT = NN$ indicates that if a word is preceded by a determiner, then it should be tagged a noun.

3. Select the single rule with the highest score. The score for a rule is the result of subtracting the number of instances whose class the rule changes to the incorrect value from the number of instances whose class the rule changes to the correct

value.

4. Append the chosen rule to the learned rule sequence and apply the rule to obtain the current corpus. The chosen rule transforms the class of the instances in the current corpus to more closely resemble their correct class.

5. Repeat Steps 2-4 (see blue loop in Figure 5.1) until error reduction is not possible, i.e., all rules have a negative score (see red arrow). During the iterative process in Steps 2-4, the class value of an instance may change several times as different rules are applied to the corpus.

6. The current sequence of rules is the final learned sequence of transformations that is applied to unseen data.

To apply the learned model to new data, the baseline system is first used to determine the initial prediction for the class of each instance in the dataset. Each rule from the learned rule sequence is then applied, in turn. After all of the rules in the sequence have been applied, the class of each instance represents the learned model's prediction.

A transformation-based machine learning algorithm has several advantages. The learned linguistic knowledge is easy to interpret by people, and the algorithm has the ability to consider many different types of features for inclusion in the rules (Samuel, Carberry, & Vijay-Shanker 1998). A disadvantage of this approach, and many others, however, is that the system is restricted to choosing rules made up of only the features with which it has been provided. It is not possible for the model to make use of knowledge that is not captured in the chosen features.

5.4 Data-Collection Exercise to Create a Corpus

To create a corpus for a transformation-based learning algorithm to use to induce a model for generating referring expressions, we conducted an online data-collection exercise. The goal of the exercise was to collect natural-sounding referring expressions that people use. 35 car repair procedures were chosen randomly from the corpus of car repair manuals[1] and divided into seven groups, each containing five procedures. Each procedure contained removal and installation subprocedures. The structure of the procedures was flat, i.e., none of the instruction steps contained embedded subprocedures. Noun phrase referring expressions contained in the procedures were replaced by a pull down menu containing two options: the full noun phrase and *other*. A partial repair procedure used in the data-collection exercise is shown in Figure 5.2. Subjects were encouraged to shorten noun phrase referring expressions as much as possible without introducing ambiguity.[2] They were told to assume that the object being referred to by the noun phrase they were given was correct, i.e., the procedures already contained the correct content. Their job was to determine how best to communicate this content. Choosing the first option in the pull down menu indicated that the full noun phrase was appropriate. If subjects chose *other* from the menu, they were prompted to type in an alternate, shortened form of the full noun phrase.

A total of 35 subjects[3] participated in the study. Subjects were randomly divided

[1] The car repair procedures in the corpus, such as the one shown in Figure 2.1, contained redundant referring expressions. Therefore, the referring expressions contained in them could not be used to induce models for generating natural-sounding referring expressions in documents.

[2] The full set of instructions that subjects were given, along with the example procedure they practiced on, can be found in Appendix C.

[3] Subjects were graduate students and college graduates aged 25-30.

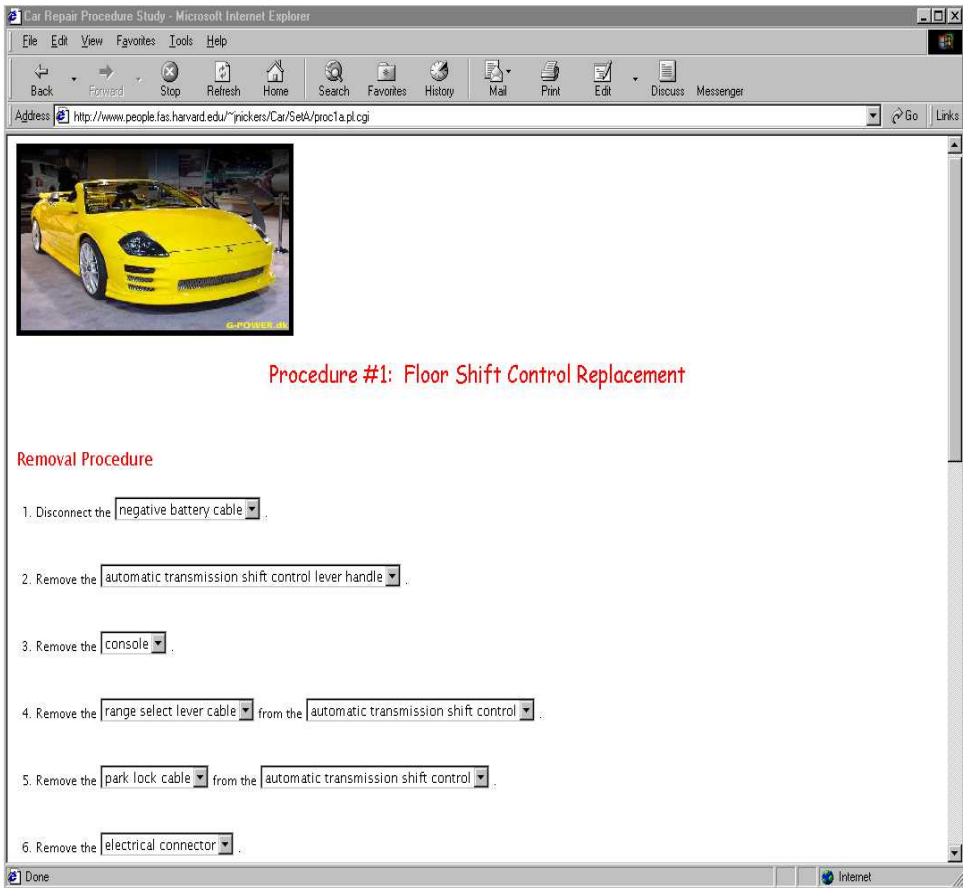


Figure 5.2: Partial Repair Procedure Used in Online Data-Collection Exercise.

into seven groups of five subjects. Each group of subjects made judgements on the referring expressions in a different procedure set. The referring-expression judgements that each subject made were logged. The data-collection exercise resulted in the collection of five judgements for each of the 500 noun phrase referring expressions contained in the procedures.

5.5 Referring-Expression Generation in the Paradigm of Transformation-Based Learning

Input to a transformation-based learning algorithm includes the following: 1) a corpus of instances, 2) the correct class of each instance in the corpus, 3) a baseline system to derive an initial hypothesis for the class of each instance, and 4) rule templates from which the algorithm creates rules for updating the class of instances. Each of the 1760 words contained in the 500 noun phrase referring expressions contained in the procedures used in the online data-collection exercise was taken to be an instance in the corpus. To arrive at the gold-standard class for each instance, we determined the referring expressions that the five subjects who made judgements on it chose most frequently. This computation revealed two interesting observations. First, pronouns were used very rarely, even though subjects were free to use them. A pronoun was never the most frequently chosen expression used to refer to an object. Also, the pronouns that were used were only used to refer to an object that had already been mentioned in the same instruction step, i.e., intrasententially. Possibly pronouns, which are used to refer to the object in focus (Grosz, Joshi, & Weinstein 1995), are inappropriate since in a car repair procedure, focus shifts quickly. A pronoun might mislead a reader to think that a particular object is in focus. Second, all of the most frequently chosen referring expressions consisted of a contiguous subsequence of words from the full noun phrase. The last word of the subsequence was always the last noun of the full noun phrase. For example, for the full noun phrase *range select lever cable*, the non-contiguous subsequence *range lever cable* was never

the most frequent selection.

Each instance in the corpus was assigned one of two possible classes: O (Outside) or I (Inside).[4] The class O indicates that the referring expression that the subjects chose most frequently did not contain this word. All words included in the referring expression that subjects chose most frequently was assigned the class I. For instance, if *shift control* is the most frequently chosen expression to refer to the automatic transmission shift control, the gold-standard class for each instance would be as follows: automatic-O, transmission-O, shift-I, and control-I. The baseline system used to assign an initial class to each instance assigns each instance the class I, indicating that the initial guess of the system is to use the full noun phrase to refer to all entities.

The 35 features contained in the rule templates for updating the class of instances are divided into seven categories: identifier (Table 5.1)[5], subsequence (Table 5.2), type (Table 5.3), coreference (Table 5.4), syntax (Table 5.5)[6], class (Table 5.6), and miscellaneous (Table 5.7).[7] Subsequence, coreference, and type features encode recency since last mention. Subsequence features are used to determine whether an object has already been referred to using the same prefix as the current one. A prefix is defined as the contiguous set of words beginning with the first word in the noun phrase[8] (excluding determiners) and ending with the current word. Type

[4]The method we use to assign a class to each instance is similar to the one used to apply transformation-based learning to text chunking (Ramshaw & Marcus 1995).

[5]In these feature tables, N and NP are used as abbreviations for noun and noun phrase, respectively. Not all of features may be relevant for a given instance. For example, the feature *REDUCE_1* does not apply to the last word in a noun phrase.

[6]Syntactic positions include the following: object, object of a preposition, object within a relative clause, and object of a preposition within a relative clause.

[7]A domain expert was used to solicit a list of approximately ten compound noun phrases for the car repair domain such as *access hole* and *control arm*.

[8]Noun phrase boundaries are encoded in the corpus.

Feature	*Meaning*
WORD	word itself

Table 5.1: Identifier Feature Used in Rule Templates.

Feature	*Meaning*
*NP_DIST_SAME_SUBSEQ**	#NPs since last use of NP with current prefix
*STEP_DIST_SAME_SUBSEQ**	#Instruction steps since last use of NP with current prefix
BINARY_SAME_SUBSEQ_PROC	Has NP with current prefix already been used in current car repair procedure?
BINARY_SAME_SUBSEQ_SUBPROC	Has NP with current prefix already been used in current subprocedure?
BINARY_SAME_SUBSEQ_SAME_STEP	Has NP with current prefix already been used in current instruction step?

Table 5.2: Subsequence Features Used in Rule Templates.

features are used to determine the recency of different objects of the same type. To automatically approximate this task, noun phrases that contain the same head noun[9] but at least one different word are considered objects of the same type. For example, *negative battery cable* and *positive battery cable* refer to different types of cables. Coreference features encode whether the object referred to by the noun phrase that the current word is a part of has been mentioned previously. Subsequence, type, and coreference features are motivated by prior referring-expression generation work that keeps track of when entities in the discourse were last mentioned and whether or not an entity needs to be distinguished from other entities of the same type to generate an appropriate referring expression (McCoy & Strube 1999; Reiter & Dale 2000, iter alia).

[9]The last word in a noun phrase is the head noun of the phrase.

Feature	*Meaning*
*NP_DIST_SAME_TYPE**	#NPs since last use of NP with same head N but different word as NP current word is part of
*STEP_DIST_SAME_TYPE**	#Instruction steps since last use of NP with same head N but different word as NP current word is part of
BINARY_SAME_TYPE_PROC	Has NP with same head N but different word already been used in current car repair procedure?
BINARY_SAME_TYPE_SUBPROC	Has NP with same head N but different word already been used in current subprocedure?

Table 5.3: Type Features Used in Rule Templates.

Feature	*Meaning*
*NP_DIST_COREF**	#NPs since last use of NP current word is part of
*STEP_DIST_COREF**	#Instruction steps since last use of NP current word is part of
BINARY_COREF_PROC	Has NP current word is part of already been used in current car repair procedure?
BINARY_COREF_SUBPROC	Has NP current word is part of already been used in current subprocedure?
BINARY_INTERVENE_SAME_TYPE_PROC	Has NP with same head N but different word been used since most recent use of NP this word is part of in current car repair procedure?
BINARY_INTERVENE_SAME_TYPE_SUBPROC	Has NP with same head N but different word been used since most recent use of NP this word is part of in current subprocedure?

Table 5.4: Coreference Features Used in Rule Templates.

Feature	*Meaning*
PREV_SYNTAX	Syntactic position of last use of NP this word is part of
CUR_SYNTAX	Syntactic position of NP current word is part of
VERB	Main verb of current instruction step
BINARY_HEAD	Is current word head noun?
POSITION	Ordinal position of current word in NP it is part of

Table 5.5: Syntax Features Used in Rule Templates.

Feature	*Meaning*
REDUCE_-5	Current class of word 5 words to left in NP current word is part of
REDUCE_-4	Current class of word 4 words to left in NP current word is part of
REDUCE_-3	Current class of word 3 words to left in NP current word is part of
REDUCE_-2	Current class of word 2 words to left in NP current word is part of
REDUCE_-1	Current class of word 1 word to left in NP current word is part of
REDUCE	Current class of current word
REDUCE_1	Current class of word 1 word to right in NP current word is part of
REDUCE_2	Current class of word 2 words to right in NP current word is part of
REDUCE_3	Current class of word 3 words to right in NP current word is part of
REDUCE_4	Current class of word 4 words to right in NP current word is part of
REDUCE_5	Current class of word 5 words to right in NP current word is part of

Table 5.6: Class Features Used in Rule Templates.

Feature	*Meaning*
TITLE	Is current word present in title of current car repair procedure?
COMPOUD	Is current word part of a compound NP?
*TF_IDF**	TF-IDF weight (Salton & Buckley 1988) of current prefix

Table 5.7: Miscellaneous Features Used in Rule Templates.

```
for each num_bins (0 2 3)
  do { for each feature_set (WORD NO_WORD)
       do { for train_num ← 1 to 5
            do { learned_rules ← LEARN(train_num, num_bins, feature_set)
                 for each prune (.001 .0025 .005 .01 .02 .025 .05 .1 .15 .2 .25 NA)
                 do { pruned_rules ← PRUNE_RULES(prune, learned_rules)
                      validation_num ← train_num
                      TEST(validation_num, pruned_rules) } } } }
```

Figure 5.3: Learning and Validating Models for Referring-Expression Generation.

The triggering context of each rule template is made up of at most three features.[10] In creating each rule, the system considered 6,000-7,000 rule templates (depending on the value of the *feature_set* parameter discussed in Section 5.6), each of which had to be instantiated with all possible values for the features in the template to form a candidate rule.

5.6 Developing Models for Referring-Expression Generation

Figure 5.3 shows the algorithm used to learn and validate models for referring-expression generation using transformation-based learning.[11] The learned models automatically generate noun phrase referring expressions. The algorithm uses 75% (1320 words) of the corpus for training and validation. The remaining 25% (440 words) is used to test the final model. The training and validation set is further

[10] More than three features proved to be too expensive computationally. Methods have been suggested for optimizing the rule consideration process to make it more tractable (Ramshaw & Marcus 1995).

[11] We used an existing transformation-based learning toolkit (Ngai & Florian 2001).

divided into five equal parts to allow for 5-fold cross-validation (Weiss & Kulikowski 1991). The validation set associated with a given training set is labeled with the same number. For example, training set 1 contains the first four folds, and validation set 1 contains the fifth fold.

The value of three parameters were adjusted to create a total of 72 parameter settings: $feature\_set$, num_bins, and $prune$. $feature\_set$ has two possible values. A value of $WORD$ indicates that the $WORD$ identifier feature is included in the rule templates. A value of NO_WORD indicates that the identifier feature is **not** used. num_bins refers to the number of bins used for discretizing select recency features,[12] i.e., features of type subsequence, type, and coreference. Discretized recency features are followed by an asterisk in the feature tables presented in Section 5.5. A num_bins value of 0 indicates that discretization was not performed. For $num\_bins = 2$, distances of one and two were assigned the bin value 1, and distances of three or more were assigned the bin value 2. For $num\_bins = 3$, we determined cutoffs for the three bins by inspecting a histogram of the possible values for each discretized feature. The bin number for a discretized feature is an attempt to draw distinctions between levels of accessibility of previously mentioned entities (Grosz & Sidner 1986). Entities mentioned recently in a document are more accessible than other previously mentioned entities. The final parameter, $prune$,[13] is used to identify rules which characterize the regularity of the car repair domain, in general, as opposed to addressing the peculiarities that may be present in the data used to train the model. The value of $prune$ represents the p value resulting from a one-tailed t test to assess whether the

[12] For all values of bin_size, TF_IDF was discretized using ten bins. Bin cutoffs were determined by inspecting a histogram.

[13] $prune = NA$ indicates that no pruning is performed.

underlying proportion of class values a rule correctly predicts is greater than 50% of the total number of instances to which the rule applies. PRUNE_RULES performs the statistical test[14] and returns the ordered set of rules for which the p value resulting from the statistical test is less than *prune*. For each *num_bins* and $feature\_set$, the algorithm in Figure 5.3 calls LEARN to learn a sequence of rules. For each value of *prune*, PRUNE_RULES then prunes the learned rule sequence. Finally, TEST applies each pruned rule sequence. This process is performed five times, each time using a different fold as the held-out one.

5.7 Results on the Validation Sets

To calculate performance metrics for each parameter setting, we compared the referring expressions generated by the learned model to those chosen by the subjects in the data-collection exercise and computed average accuracy and root mean square (RMS) error over the five validation sets. Accuracy is measured by comparing the noun phrases the models generate to the those most frequently chosen by subjects. A referring expression generated by the learned models is counted as correct only if it exactly matches the gold-standard referring expression. RMS error measures the deviation in length (in number of words) of the referring expression generated by the model from the gold-standard referring expression.[15] For instance, an RMS error of 1.5 indicates that, on average, referring expressions generated by the model are

[14] Prior work in the area of machine learning has used statistical tests to sort rules based on their accuracy and reliability (Yarowsky 1994; Clark & Niblett 1989, inter alia).

[15] Before calculating RMS error for the learned model, it was verified that all referring expressions contained a contiguous sequence of nouns from the full noun phrase, with the head noun from the full noun phrase being the right-most noun in the predicted referring expression. If this constraint were not enforced, the RMS error rate for the learned model would be incorrectly deflated.

prune	*feature_set*	
	NO_WORD	*WORD*
.001	67.55% ± 2.50% 0.93 ± 0.21	67.55% ± 2.50% 0.93 ± 0.21
.0025	68.02% ± 4.13% 0.94 ± 0.19	64.25% ± 5.85% 1.09 ± 0.28
.005	67.62% ± 4.14% 0.94 ± 0.19	63.79% ± 5.46% 1.09 ± 0.28
.01	67.62% ± 4.14% 0.92 ± 0.21	63.79% ± 5.46% 1.07 ± 0.31
.02	65.57% ± 5.69% 1.04 ± 0.35	61.73% ± 5.03% 1.19 ± 0.36
.025	65.57% ± 5.69% 1.04 ± 0.35	61.73% ± 5.03% 1.19 ± 0.36
.05	65.02% ± 4.73% 1.04 ± 0.34	61.17% ± 5.21% 1.19 ± 0.37
.1	65.02% ± 4.73% 1.04 ± 0.34	61.17% ± 5.21% 1.19 ± 0.37
.15	65.02% ± 4.73% 1.02 ± 0.34	61.17% ± 5.21% 1.17 ± 0.38
.2	65.05% ± 4.73% 1.02 ± 0.34	61.17% ± 5.21% 1.17 ± 0.38
.25	65.02% ± 4.73% 1.02 ± 0.34	61.17% ± 5.21% 1.17 ± 0.38
NA	64.81% ± 4.82% 1.02 ± 0.23	59.19% ± 4.35% 1.23 ± 0.31

Table 5.8: $num\_bins = 0$: Accuracy and RMS Error Results Averaged over Five Validation Sets.

within 1.5 words in length, either longer or shorter, of the gold-standard referring expressions. Tables 5.8, 5.9, and 5.10 show the performance of the rule sequences for parameter settings in which $num\_bins = 0$, $num\_bins = 2$, and $num\_bins = 3$, respectively. The average accuracy of the baseline model on the five validation sets is 61.09% ± 9.69%, and the average RMS error is 1.56 ± 0.49.

Three interesting observations can be gleaned from the performance tables. 1)

prune	*feature_set*	
	NO_WORD	*WORD*
.001	66.54% ± 4.58% 0.92 ± 0.14	66.54% ± 4.58% 0.92 ± 0.14
.0025	69.60% ± 2.82% 0.84 ± 0.15	69.60% ± 2.82% 0.84 ± 0.15
.005	70.68% ± 1.97% 0.93 ± 0.34	70.68% ± 1.97% 0.93 ± 0.34
.01	70.68% ± 1.97% 0.93 ± 0.34	70.68% ± 1.97% 0.93 ± 0.34
.02	67.32% ± 4.80% 1.00 ± 0.32	67.32% ± 4.80% 1.00 ± 0.32
.025	67.32% ± 4.80% 1.00 ± 0.32	67.32% ± 4.80% 1.00 ± 0.32
.05	68.56% ± 2.85% 1.14 ± 0.32	68.56% ± 2.85% 1.14 ± 0.32
.1	68.56% ± 2.85% 1.14 ± 0.32	68.56% ± 2.85% 1.14 ± 0.32
.15	68.09% ± 4.67% 1.17 ± 0.35	68.09% ± 4.67% 1.17 ± 0.35
.2	68.09% ± 4.67% 1.17 ± 0.35	68.09% ± 4.67% 1.17 ± 0.35
.25	68.09% ± 4.67% 1.17 ± 0.35	68.09% ± 4.67% 1.17 ± 0.35
NA	65.25% ± 6.12% 1.16 ± 0.32	69.00% ± 3.96% 1.10 ± 0.26

Table 5.9: *num_bins* = 2: Accuracy and RMS Error Results Averaged over Five Validation Sets.

prune	*feature_set*	
	NO_WORD	*WORD*
.001	65.07% ± 4.26% 1.01 ± 0.17	65.07% ± 4.26% 1.01 ± 0.17
.0025	65.07% ± 4.26% 1.01 ± 0.17	65.07% ± 4.26% 1.01 ± 0.17
.005	65.19% ± 3.90% 1.01 ± 0.16	65.19% ± 3.90% 1.01 ± 0.16
.01	65.19% ± 3.90% 0.99 ± 0.19	65.19% ± 3.90% 0.99 ± 0.19
.02	62.56% ± 3.80% 1.02 ± 0.15	62.55% ± 3.80% 1.02 ± 0.15
.025	62.55% ± 3.80% 1.02 ± 0.15	62.55% ± 3.80% 1.02 ± 0.15
.05	63.24% ± 4.89% 0.97 ± 0.18	63.24% ± 4.89% 0.97 ± 0.18
.1	63.24% ± 4.89% 0.97 ± 0.18	63.24% ± 4.89% 0.97 ± 0.18
.15	65.46% ± 4.84% 0.96 ± 0.16	62.19% ± 8.86% 1.03 ± 0.31
.2	65.46% ± 4.83% 0.96 ± 0.16	62.19% ± 8.86% 1.03 ± 0.31
.25	65.46% ± 4.84% 0.96 ± 0.16	62.19% ± 8.86% 1.03 ± 0.31
NA	65.41% ± 3.41% 1.03 ± 0.19	61.84% ± 8.00% 1.10 ± 0.26

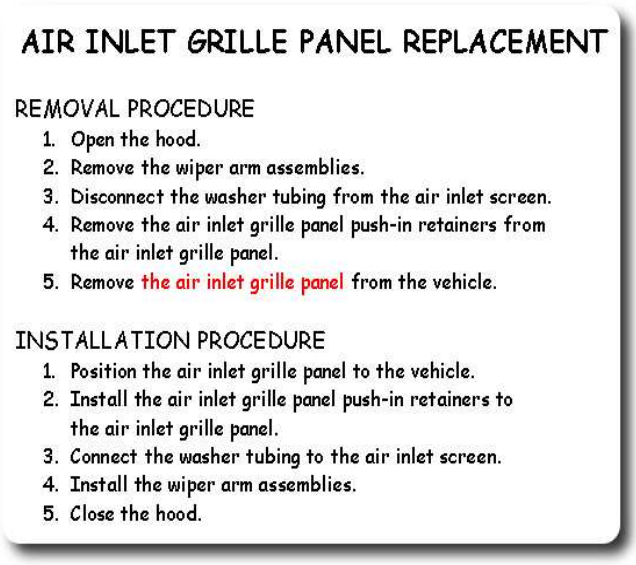
Table 5.10: *num_bins* = 3: Accuracy and RMS Error Results Averaged over Five Validation Sets.

Rule	*Triggering Context*	*New Class*
1	$NP\_DIST\_SAME\_SUBSEQ = 1 \wedge$ $BINARY\_SAME\_SUBSEQ\_SUBPROC = 1 \wedge$ $REDUCE\_2 = I$	O
2	$STEP\_DIST\_COREF = 1 \wedge$ $BINARY\_COREF\_SUBPROC = 1 \wedge$ $BINARY\_HEAD = 0$	O
3	$TF\_IDF = 5 \wedge TITLE = 1 \wedge$ $REDUCE\_1 = I$	I

Table 5.11: A Learned Rule Sequence for $num\_bins = 2$, $feature\_set = NO\_WORD$, and $prune = .0025$.

In general, for $prune > .01$, performance degrades. These rule sequences overfit the training data, and, therefore, do not generalize well to the unseen instances in the validation sets. 2) For rule sequences in which overfitting is not a factor, models developed using $num\_bins = 2$ perform best. 3) The identifier feature does not contribute to improving performance. For this reason, most learned rule sequences did not make use of the identifier feature when it was available, i.e., for parameter settings in which $feature\_set = WORD$, hence the equivalent performance of rule sequences induced for $feature\_set = WORD$ and $feature\_set = NO\_WORD$, holding all other parameters equal. Rule sequences that make use of the identifier feature are not as general and do not perform as well as the rule sequences developed using the counterpart parameter setting that does not. For example, the performance of the rule sequences developed for $num\_bins = 0$ and $prune = .005$ is better when $feature\_set = NO\_WORD$.

Table 5.11 presents an example learned rule sequence for the following parameter setting: $num\_bins = 2$, $feature\_set = NO\_WORD$, and $prune = .0025$. Tables 5.2–5.7 can be used to determine the meaning of the rules. Rule 1, for example, indicates



AIR INLET GRILLE PANEL REPLACEMENT

REMOVAL PROCEDURE

1. Open the hood.
2. Remove the wiper arm assemblies.
3. Disconnect the washer tubing from the air inlet screen.
4. Remove the air inlet grille panel push-in retainers from the air inlet grille panel.
5. Remove the air inlet grille panel from the vehicle.

INSTALLATION PROCEDURE

1. Position the air inlet grille panel to the vehicle.
2. Install the air inlet grille panel push-in retainers to the air inlet grille panel.
3. Connect the washer tubing to the air inlet screen.
4. Install the wiper arm assemblies.
5. Close the hood.

Figure 5.4: Car Repair Procedure Used to Demonstrate Application of Learned Rule Sequence in Table 5.11.

that if 1) one of the two previous noun phrases used in the car repair procedure has the same prefix as the current word, 2) a noun phrase with the same prefix as the current word is used in the current subprocedure, and 3) the current class of the word located two words to the right of the current word in the noun phrase it is part of is currently assigned the class I, the current instance should be assigned the class O.

If the current car repair procedure is the air inlet grille panel replacement procedure shown in Figure 5.4, and the learned sequence in Table 5.11 is used to generate a referring expression for the *air inlet grille panel* object referred to in Step 5 of the removal procedure, the baseline system is first used to assign an initial class value to each word in the noun phrase. This initial class assignment is shown in Table 5.12. The initial referring expression the baseline system generates is the full noun phrase, *air inlet grille panel*. The first and second rules in the learned sequence demonstrate the need to drop words when objects are being referred to in the locality of similar

Words in Noun Phrase	*Baseline Class Assignment*	*After Rule 1*	*After Rule 2*	*After Rule 3*
air	I	O	O	O
inlet	I	O	O	O
grille	I	I	O	O
panel	I	I	I	I

Table 5.12: Class Assignment Before and After Application of Learned Rule Sequence in Table 5.11.

objects. Since the prefixes *air* and *air inlet* are used in the previous noun phrase (*air inlet grille panel* in Step 4), which is contained in the current subprocedure, and because the current class of *grille* and *panel* is I, the application of Rule 1 changes the class of *air* and *inlet* as shown in the table. The current referring expression after the application of the first rule is *grille panel*. The application of Rules 2 and 3 proceeds in a similar manner. The second rule applies to *air*, *inlet*, and *grille*. It does not apply to *panel*, since *panel* is the head noun of the noun phrase. The class of *grille* changes to O. The current referring expression is now *panel*. Finally, the third rule does not apply to any of the words in the noun phrase. The discretized TF-IDF weights of the prefixes are as follows: air:3, air inlet:2, air inlet grille:3, and air inlet grille panel:5.[16] *Panel* satisfies the first two conjuncts of the rule but fails to satisfy the last rule conjunct. The final referring expression generated by the rule sequence is *panel*.

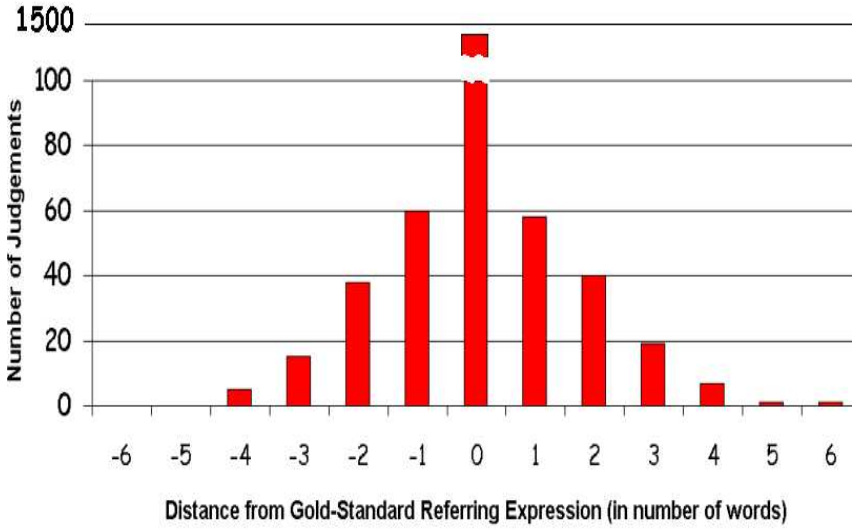


Figure 5.5: Distance from Gold-Standard Referring Expression for Most Frequent Judgements in Data-Collection Exercise. (Position 0 Has a Value of 1445 Judgements.)

5.8 Developing the Final Learned Model

To determine the parameter setting used to develop the final learned model, we optimized the RMS error rate evaluation metric, the metric that people optimize when making referring-expression judgments. Figure 5.5 shows that, if the decision of the subjects differs from the gold-standard referring expression, those referring expressions that are only off by a few words are preferred. When subjects chose a referring expression that is different from the gold standard, for 118 out of 244 referring expressions, the chosen expression is within one word, either longer or shorter, of the gold standard. For 78 referring expressions, the chosen one is within two words of the gold standard. The parameter setting that led to the development of rule

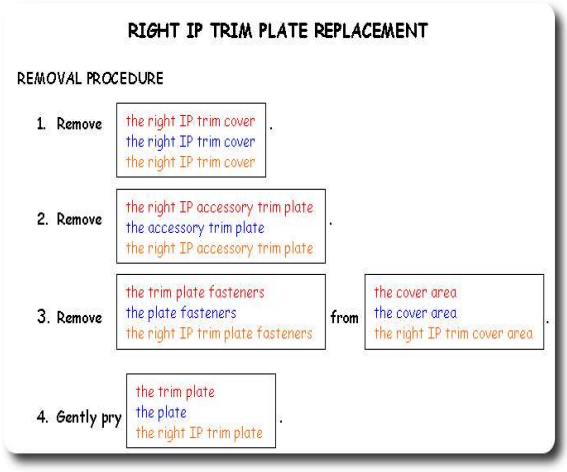
[16] A discretized TF-IDF weight of 5 indicates that, more so than the other prefixes, *air inlet grille panel* has a high frequency in the car repair procedure in Figure 5.4 and a low frequency in the entire collection of car repair procedures.

Rule	*Triggering Context*	*New Class*
1	$BINARY\_SAME\_SUBSEQ\_SUBPROC = 1 \wedge$ $BINARY\_HEAD = 0 \wedge$ $REDUCE\_\text{-}3 = -$	O
2	$NP\_DIST\_COREF = 2 \wedge$ $REDUCE\_\text{-}2 = - \wedge$ $REDUCE\_1 = I$	I
3	$TF\_IDF = 2 \wedge REDUCE\_\text{-}2 = -$	I
4	$NP\_DIST\_COREF = - \wedge$ $BINARY\_SAME\_SUBSEQ\_SAME\_STEP = 0 \wedge$ $COMPOUND = 1$	I

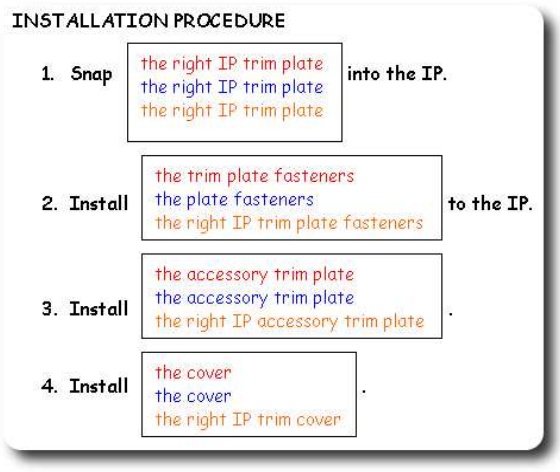
Table 5.13: Final Learned Rule Sequence for $num\_bins = 2$, $feature\_set = NO\_WORD$, and $prune = .0025$.

sequences with the lowest average RMS error rate is $num\_bins = 2$, and $prune = .0025$. Because inclusion of the identifier feature may lead to rule sequences that are not as general, we chose $feature\_set = NO\_WORD$. The average RMS error rate of the rule sequences induced using this parameter setting on the validation sets, 0.84 ± 0.15, is significantly lower than that of the baseline system, 1.56 ± 0.49 ($p < 0.02$).

To develop the final learned model for referring-expression generation, we trained the model using 75% of the data collected from the data-collection exercise, the portion used for training and validation. To determine how well the model generalizes to unseen data, the remaining 25% of the data was used to test it. Table 5.13 presents the final learned rule sequence. A feature value of "-" indicates that the feature does not apply to the instance. The first rule indicates that if 1) a word is one of the first three words contained in a noun phrase, 2) it is not the head of the noun phrase, and 3) another noun phrase with the same prefix has already been mentioned in the current subprocedure, this word should be dropped. Rules 2-4 represent conditions for retaining a word in a noun phrase. Some of the conditions verified before retaining



Removal Procedure



Installation Procedure

Figure 5.6: Comparison of Gold-Standard Referring Expressions (Red) with Those Generated by the Learned Model (Blue) and the Baseline System (Orange).

a noun include checking that the previous mention of the noun phrase is far away (Rule 2) and confirming that the noun phrase has not already been mentioned in the current procedure (Rule 4). The RMS error rate of the baseline system on the held-out test set is 1.38. For the learned rule sequence, it is 0.81.[17]

5.9 Analysis of the Predictions Made by the Final Learned Model

Figure 5.6 compares the gold-standard referring expressions with those generated by the learned model and the baseline system. The first referring expression in each box, the red one, is the gold standard. The second referring expression, the blue

[17]The accuracy of the baseline system on the held-out test set is 59.69%; the accuracy of the learned model, 64.34%.

one, is the one generated by the learned model; and the third, the orange one, the baseline system. The predictions of the learned model differ from the gold-standard referring expression for four of the nine referring expressions contained in the car repair procedure. The model predicts that, in removal procedure Step 2, the full noun phrase *right IP accessory trim plate* can be reduced to *accessory trim plate*. The gold-standard referring expression, however, is the full noun phrase. It is possible that more than one referring expression may be acceptable in a given context. In this case, for example, since the right IP accessory trim plate is being referred to in a context in which the right IP trim cover has been mentioned in the previous instruction step, reducing the full noun phrase is likely to be acceptable. Also, the subjects judged the reduction predicted by the model in this step to be licensed in a very similar context, namely Step 3 of the installation procedure. In Steps 3 and 4 of the removal procedure and Step 2 of the installation procedure, the model correctly predicts that it is possible to reduce the referring expressions. The reduction chosen by the model in each case was chosen by subjects in the data-collection exercise, yet it was not the most frequently chosen expression. Perhaps some subjects believed *trim plate* to be a compound. The fact that they reduced *right IP trim cover* to *cover* in certain contexts, however, indicates that *plate* and *plate fasteners* may be acceptable reductions. The alternative provided by the baseline in these contexts is redundant. For example, Step 3 of the removal procedure states: Remove the *right IP trim* plate fasteners from the *right IP trim* cover area.

Once the learned model has been used to generate referring expressions, a final post-processing step is needed to ensure that the output text produced unambiguously

identifies the correct entities to the reader. For all cases in which two distinct entities of the same type have the same full form, difference words must be included in the referring expression generated by the learned model. For instance, in Figure 5.6, the entity of type *right IP trim cover* referred to in Step 1 of the removal procedure is the malfunctioning one, whereas the one referred to in Step 4 of the installation procedure is the new one. Difference words such as *old* and *new* must be prepended to the series of nouns predicted for inclusion in the referring expression generated by the model. Following post-processing, Step 4 of the removal and Step 1 of the installation procedures would read *Gently pry the* **old** *plate* and *Snap the* **new** *right IP trim plate into the IP*, respectively.

5.10 Possible Extensions

The data-collection studies presented in this chapter have shown that some amount of noun phrase reduction is desirable in the domain of car repair procedures. There are, however, some extreme circumstances under which a conservative algorithm that always generates the full noun phrase, regardless of the context, may be preferable. Three factors that may be helpful in identifying these circumstances include the following: skill level of the repair-person, task being performed, and physical proximity and nature of car parts available. An inexperienced repair-person may require fewer reduced noun phrases. Reduced forms are likely to be confusing to a novice repair-person who may not be as familiar with car parts and who may need more guidance. A more experienced repair-person, on the other hand, may find the full forms redundant and prefer reduced forms.

Full forms may also be preferred if the repair-person is skimming certain steps of a manual. Reduced forms may lead to slower comprehension time, as the repair-person may have to read steps preceding the steps of interest in order to set-up the context within which reduced referring expressions need to interpreted.

The context used to generate referring expressions not only depends on the document context, but also on the physical context (Grosz 1977). Two elements of the physical context that need to be taken into account to generate appropriate referring expressions are proximity and nature of objects available to the person performing a task. For example, if repair-people are choosing from all possible car parts as they perform an installation procedure, more specific referring expressions, namely full noun phrases, are likely to be needed to allow them to effectively distinguish desired car parts from all other car parts. If, however, they have only the parts mentioned in the removal procedure in front of them as they perform the installation procedure, reduced noun phrases are likely to be desirable, since they are choosing from a smaller set of car parts.

To address the possible need for documents with full noun phrases, electronic car repair manuals could be equipped with a toggle that allows the repair-person to switch between generating reduced noun phrases and generating full noun phrases. Alternatively, the system could construct user models that would allow it to determine the types of referring expressions appropriate for a given repair-person.

There are several alternative evaluation methods that would be interesting to examine. We could compare a system that uses the learned model for referring-expression generation to systems that use alternative methods for generating referring

expressions. The evaluation would investigate the comprehensibility and usability of generated documents for different types of tasks. Reading time and eye movement could be measured to gauge comprehension. We could also conduct a study in which subjects are provided with two types of car repair procedures. Each noun phrase in the two types of procedures is replaced by a pull down menu containing the full noun phrase, the referring expression predicted by the learned model, and *other*. One type of procedure would have the full noun phrase chosen initially, while the other one would have the referring expression predicted by the learned model chosen as the default. Choosing the *other* object would result in a prompt to type in an alternate referring expression. The subjects would be directed to make the procedures sound coherent. We would count the number of edits subjects made while interacting with the two types of procedures and expect that they would make significantly more edits when specifying referring expressions in procedures for which the initially chosen referring expression is the full noun phrase.

As Section 5.9 pointed out, in some cases, more than one referring expression may be acceptable; there is not always one correct noun referring expression for an entity. For cases in which the learned model and the baseline differ from the gold standard, we could compare the acceptability of the decisions of the learned model and the baseline system. We would hope to show that in cases when the predictions made by the learned model, the baseline system, and the gold standard differ, more often, the decisions of the learned model are judged to be acceptable, while those of the baseline are not. To arrive at such acceptability decisions, we could provide subjects with car repair procedures in which noun phrases are replaced by a pull down menu containing

three options: 1) full noun phrase (prediction of baseline system), 2) prediction of the learned model, and 3) gold standard. Subjects would rank, with the option of imposing a partial order, referring expressions in terms of their acceptability.

Further analysis of the contribution of the different types of features would be helpful for determining which features are most useful for generating referring expressions. We could also include additional features that might be informative in generating referring expressions in car repair procedures containing nested procedures, like the repair procedure in Figure 2.1. In extending the learned models to generate referring expressions in nested procedures, features that take discourse structural relations into account (McCoy & Strube 1999; Dale & Reiter 1995, inter alia) are likely to be informative. For example, whether or not an object has been referred to by an noun phrase with the same prefix in the parent discourse segment may indicate that a noun phrase in the current discourse segment should be reduced.

A property of the learned models worth exploring is speaker independence. It is possible that different people prefer different types of referring expressions. We could learn models based on the decisions of different subjects and see if the learned models differ based on the person whose judgements were used for the gold-standard decisions. Prior work that used machine learning to generate referring expressions has shown that the use of speaker-specific features leads to improved performance (Jordan & Walker 2000).

Chapter 6

Conclusion and Future Research

Multilingual document production is an important problem given the need to make information accessible to people who speak many languages. MT and MNLG are two common approaches for arriving at the language-independent meaning representation that is prerequisite for producing documents in multiple languages automatically. This thesis investigated the use of a third approach, symbolic authoring. Symbolic authoring approaches have the potential to offer the best of both worlds; they strive to retain the naturalness of MT approaches and the directness of MNLG approaches. WYSIWYM (Power, Scott, & Evans 1998) is the symbolic authoring approach that this thesis examined. Since WYSIWYM uses text as part of the dialogue for communicating the content of the language-independent representation, the problem of reference, which arises due to the inherent ambiguity in natural language, must be addressed. Reference in WYSIWYM is specified more directly than in MT approaches, but a problem that remained was to make reference specification more natural. This thesis provided methods and empirical evidence for making reference

more natural in symbolic authoring approaches, such as WYSIWYM. It addressed the problem of reference in both the KB specification and document generation phases of multilingual document production.

Categorical organization and a model based on domain and lexical knowledge were used to communicate about new entities. These methods were shown to outperform traditional methods used for referring to new entities in KB specification. To extend a system to handle repeated mention of entities during KB specification, the thesis described several properties system designers must consider. For ranking coreference candidates, discourse structure and genre features were shown to be useful. The system must also realize the content of the specified KB in a coherent natural-language document. The coherency of such documents relies heavily on the use of natural-sounding referring expression. To this end, the thesis described models induced using a transformation-based learning algorithm. The models generate referring expressions that more closely match the decisions of people, when compared to an algorithm that always generates full noun phrases.

6.1 Future Research

There are several avenues worthy of consideration for future research in the area of reference specification in multilingual documentation production. The first involves demonstrating useful properties, namely language and domain independence, of the reference mechanisms developed in this thesis. Second, the reference mechanisms could be extended to handle a different kind of reference–reference in diagrams. Finally, changing the way in which WYSIWYM-based systems are used for authoring

documents would result in new and challenging reference problems.

The first goal for future research is to show that the reference mechanisms described in this thesis are portable to other languages and domains. Keeping the domain fixed, the domain and lexical model, which uses language-independent verb-object associations, could be used directly for specifying new objects in another language. If a corpus of documents in another language became available for retraining or updating the domain and lexical model, a parser and tagger for the new language would be required. In addition, the syntactic patterns for extracting verbs and noun phrases would have to be rewritten if the word order of the new language differed from that of English, namely SVO (Subject-Verb-Object). The coreference mechanisms for ranking already mentioned menu options, too, do not make use of language-specific information. They, therefore, could be used directly for coreference specification in another language. The learned model for referring-expression generation could be used directly for languages that order constituents head-finally and observe the same modifier order as English. For languages that have the same modifier order as English and order constituents head-initially, the model could easily be modified to place all modifying elements after, as opposed to before, the head noun. For languages that order modifiers differently than English or those that do not order constituents head-initially or head-finally, however, the model would need to be retrained. Some languages, for example, order modifiers within a noun phrase based on the precedence of different types of modifiers. Even though models for these types of languages would need to be retrained, it is possible that the same set of rule template features used for referring expression generation in English could be used as

a starting point, with the addition of features characterizing the peculiarities (such as precedence order of modifiers) of the new language.

It would be interesting to characterize those domains in which the reference mechanisms are useful. For other domains containing informative verbs, the domain and lexical model is likely to be useful. It would, however, need to be retrained with a corpus of documents from the new domain. After observing repeated occurrences of informative verbs, the domain and lexical model would learn relevant verb-object associations for the new domain. The genre features used by the coreference mechanisms are domain dependent. If the new domain has genre features of its own, these features could be used in place of those specific to the domain of car repair manuals. For reference generation in domains with referring expressions containing many modifiers and few pronouns, it is likely that the model for reference generation could be used directly. The rules do not contain any features that make them specific to the domain of car repair manuals. Another interesting problem would be to examine the applicability of the reference mechanisms to more varied domains. For example, the mechanisms could be incorporated in an interactive tool for writing computer programs (Reps 1984). To prove useful, the context used to make predictions concerning the program elements likely to be chosen next would have to be revised. In this domain, structural elements, such as function or control structure blocks, are likely to be useful in predicting the variables that a programmer is likely to use again. For more varied domains, the learned model for reference generation would most likely have to be retrained using data from the new domain. It is likely, though, that many of the features used for the domain of car repair manuals could be reused.

An extension to the reference mechanisms presented in this thesis would be to allow them to support reference to diagrams (van Deemter & Power 1999), in addition to reference to entities. Procedural texts often contain diagrams to clarify or reinforce instructions communicated in text. The text refers to the parts of the diagram that pertain to the textual description. Interesting challenges include facilitating reference to parts of diagrams in the user interface and generating the appropriate text to unambiguously refer to the desired parts of the diagram.

The final two directions for future work are a result of altering the way in which documents are specified using WYSIWYM to allow domain experts to produce multilingual documentation more efficiently. In these specification methods, experts take on more of a correcting or verifying role, as opposed to purely authoring. First, the limitation of the amount of initiative the system is given coupled with the increased information regarding the predictions of experts' actions gained as a result of the mechanisms described in this thesis suggest a natural direction for future research–giving the system more initiative and, thus, making the approach a mixed-initiative symbolic authoring one. The lexical model could compute a probability representing the likelihood that each object would be selected following a given verb. If one object is much more likely than all of the others, the system could add this new object to the KB automatically. For cases of coreference, domain and lexical knowledge could be used to rule out likely coreference candidates, as judged by the discourse structure. Again, if one candidate is significantly more likely than the others, the system could specify this candidate without seeking approval from experts. Using knowledge of verbs that are opposites, such as *disconnect* and *connect*,

the system could attempt to automatically specify inverse procedures in domains in which these types of procedures are required. For example, in the car repair domain, after experts have specified a removal procedure, the system could attempt to automatically specify the installation procedure. Instances of coreference and other automatically specified material could be verified and corrected by experts through the use of a WYSIWYM-based system equipped with the reference mechanisms described in this thesis. Alternatively, instead of having the system do all of the work, experts could be given more initiative at the task that they do best–writing in natural language. Due to the intractability of MT, however, there would have to be several caveats. First, experts would be given guidelines to increase the likelihood that the system would be able to correctly process their writing. They would be encouraged to write syntactically simple sentences in which the verb is the first word. Research in the area of controlled language MT has shown that short, unambiguous source language texts improve the quality of translations (Mitamura & Nyberg 1995; Bernth 1998, inter alia). Given controlled natural-language input, the WYSIWYM-based system would attempt to mold experts' natural-language specification into text generated using the constrained grammar and objects about which it is knowledgeable. For a given sentence, the system would first check to see if the verb, or one which it has evidence to believe is similar, is contained in its ontology. Text chunking algorithms (Tjong, Sang, & Buchholz 2000; Ramshaw & Marcus 1995, inter alia) could be used to identify noun phrases. Pronouns and reduced noun phrases would present an interesting challenge. A pronoun resolution algorithm (Baldwin 1997; Ng & Cardie 2002, inter alia) and heuristics could be used to resolve and expand

expressions before attempting to match them to terms contained in the system's ontology. If no match were found in the system's ontology for an argument, it would be presented as an anchor. Anchors would also be used if arguments could not be identified for a specified verb. Again, a WYSIWYM-based system could be used to correct the system's approximation of domain experts' natural-language specification.

References

[1] Asher, N., and Hajime, W. 1988. A computational account of syntactic, semantic and discourse principles for anaphora resolution. *Journal of Semantics* 6:309–344.

[2] Baldwin, B. 1997. CogNIAC: High precision coreference with limited knowledge and linguistic resources. In *Proceedings of ACL-EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts.*

[3] Bernth, A. 1998. EasyEnglish: Preprocessing for MT. In *Proceedings of CLAW.*

[4] Bird, S.; Maeda, K.; Ma, X.; Lee, H.; Randall, B.; and Zayat, S. 2002. TableTrans, MultiTrans, InterTrans, and TreeTrans: Diverse tools built on the annotation graph toolkit. In *Proceedings of LREC.*

[5] Blaheta, D., and Charniak, E. 2000. Assigning function tags to parsed text. In *Proceedings of NAACL.*

[6] Borgida, A.; Brachman, R.; McGuiness, D.; and Resnick, L. 1989. CLASSIC: A structural data model for object. In *Proceedings of SIGMOD Conference on the Management of Data.*

[7] Bouayad-Agha, N.; Power, R.; Scott, D.; and Belz, A. 2002. PILLS: Multilingual generation of medical information documents with overlapping content. In *Proceedings of LREC.*

[8] Brill, E., and Resnik, P. 1994. A transformation-based approach to prepositional phrase attachment disambiguation. In *Proceedings of COLING.*

[9] Brill, E. 1993. *A corpus-based approach to language learning.* Ph.D. Dissertation, University of Pennsylvania.

[10] Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics* 21(4):543–566.

[11] Brun, C.; Dymetman, M.; and Lux, V. 2000. Document structure and multilingual text authoring. In *Proceedings of INLG.*

[12] Caldwell, D., and Korelsky. 1994. Bilingual generation of job descriptions from quasi-conceptual forms. In *Proceedings of ANLP.*

[13] Callaway, C., and Lester, J. 2002. Pronominalization in generated discourse and dialogue. In *Proceedings of ACL.*

[14] Carbonell, J., and Brown, R. 1988. Anaphora resolution: A multi-strategy approach. In *Proceedings of COLING.*

[15] Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3(4):261–283.

[16] Clark, H. 1977. Bridging. *Thinking: Readings in Cognitive Science.*

[17] CLIM. 1994. Common Lisp Interface Manager: User Guide.

[18] Cohen, P. 1995. *Empirical Methods for Artificial Intelligence.* MIT Press.

[19] Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL.*

[20] Cristea, D.; Ide, N.; Marcu, D.; and Tablan, V. 1999. Discourse structure and coreference: An empirical study. In *Proceedings of ACL Workshop on the Relation of Discourse/Dialogue Structure and Reference.*

[21] Cristea, D.; Ide, N.; and Romary, L. 1998. Veins theory: A model of global discourse cohesion and coherence. In *Proceedings of ACL-COLING.*

[22] Dagan, I., and Itai, A. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *Proceedings of COLING.*

[23] Dale, R., and Reiter, E. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2):233–263.

[24] Day, D.; Aberdeen, J.; Hierchman, L.; Kozierok, R.; Robinson, P.; and Vilain, M. 1997. Mixed-initiative development of language processing systems. In *Proceedings of ANLP.*

[25] GenDesigner. 2004. GenDesigner 3.0. World Wide Web. http://www.gendesigner.com.

[26] General Motors. 2003. General Motors Service Information Web Pages. World Wide Web. http://service.gm.com.

[27] Gordon, P., and Hendrick, R. 1998. The representation and processing of coreference in discourse. *Cognitive Science* 22(4):389–424.

[28] Gordon, P.; Grosz, B.; and Gilliom, L. 1993. Pronoun, names, and the centering of attention in discourse. *Cognitive Science* 17(3):311–347.

[29] Grosz, B., and Sidner, C. 1986. Attention, intentions and the structure of discourse. *Computational Linguistics* 12(3):175–204.

[30] Grosz, B.; Joshi, A.; and Weinstein, S. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225.

[31] Grosz, B. 1977. *The representation and use of focus in dialogue understanding.* Ph.D. Dissertation, University of California, Berkeley.

[32] Grosz, B. 1978. Discourse analysis. In Walker, D., ed., *Understanding Spoken Language.* Elsevier Science Inc. 235–268.

[33] Hana, J. 2001. The AGILE system. *Prague Bulletin of Mathematical Linguistics* 39–67.

[34] Ide, N., and Cristea, D. 2000. A hierarchical account of referential accessibility. In *Proceedings of ACL.*

[35] Johnson, J. 2000. *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers.* Morgan Kaufmann.

[36] Jordan, P., and Walker, M. 2000. Learning attribute selection for non-pronominal expressions. In *Proceedings of ACL.*

[37] Kehler, A.; Appelt, D.; Taylor, L.; and Simma, A. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of HLT-NAACL.*

[38] Kibble, R., and Power, R. 2000. An integrated framework for text planning and pronominalisation. In *Proceedings of INLG.*

[39] Kinyon, A., and Prolo, C. 2002. Identifying verb arguments and their syntactic function in the Penn treebank. In *Proceedings of LREC.*

[40] Lappin, S., and Leass, J. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics* 20(4):535–561.

[41] Liberatore, P., and Schaerf, M. 1998. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering* 10(1):76–90.

[42] Liberatore, P., and Schaerf, M. 2000. BReLS: A system for the integration of knowledge bases. In *Proceedings of KR.*

[43] Macias, B., and Pulman, S. 1995. A method for controlling the production of specifications in natural language. *The Computer Journal* 38(4):310–318.

[44] Manning, C., and Schutze, H. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press.

[45] Martin, D. 2003. *Doing Psychology Experiments.* Wadsworth.

[46] McCoy, K., and Strube, M. 1999. Generating anaphoric expressions: Pronoun or definite description? In *Proceedings of the Workshop on the Relation of Discourse/Dialogue Structure and Reference held in conjunction with ACL.*

[47] Means, L., and Godden, K. 1996. The controlled automotive service language (CASL) project. In *Proceedings of CLAW.*

[48] Mitamura, T., and Nyberg, E. 1995. Controlled English for knowledge-based MT: Experience with the KANT system. In *Proceedings of TMI.*

[49] Ng, V., and Cardie, C. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL.*

[50] Ngai, G., and Florian, R. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL.*

[51] Nickerson, J. 2002. Controlling linguistic coreference in graphical interfaces. In *Proceedings of ACL Student Session.*

[52] Nickerson, J. 2003. Statistical models for organizing semantic options in knowledge editing interfaces. In *Proceedings of AAAI Spring Symposium Series Workshop on Natural Language Generation in Spoken and Written Dialogue.*

[53] Palm. 1996. Palm Desktop. World Wide Web. http://www.palmsource.com.

[54] Paris, C.; Vander Linden, K.; Fischer, M.; Hartley, A.; Pemberton, L.; Power, R.; and Scott, D. 1995. A support tool for writing multilingual instructions. In *Proceedings of IJCAI.*

[55] Piwek, P.; Evans, R.; Cahill, L.; and Tipper, N. 2000. Natural language generation in the MILE system. In *Proceedings of the IMPACTS in NLG Workshop.*

[56] Poesio, M.; Bruneseauz, F.; and Romary, L. 1999. The MATE meta-schedule for coreference in dialogues in multiple languages. In *Proceedings of ACL Workshop on Standards for Discourse Tagging.*

[57] Power, R.; Scott, D.; and Evans, R. 1998. What you see is what you meant: Direct knowledge editing with natural language feedback. In *Proceedings of ECAI.*

[58] Power, R. 1999. Generating referring expressions with a unification grammar. In *Proceedings of EWNLG.*

[59] Ramshaw, L., and Marcus, M. 1995. Text chunking using transformation-based learning. In *Proceedings of ACL Workshop on Very Large Corpora.*

[60] Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP.*

[61] Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems.* Cambridge University Press.

[62] Reps, T. 1984. *Generating Language-Based Environments.* MIT Press.

[63] Rich, E., and LuperFoy, S. 1988. An architecture for anaphora resolution. In *Proceedings of ANLP.*

[64] Rohde, D. 2001. TGrep2. World Wide Web. http://tedlab.mit.edu/~dr/Tgrep2.

[65] Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24(5):513–523.

[66] Samuel, K.; Carberry, S.; and Vijay-Shanker, K. 1998. An investigation of transformation-based learning in discourse. In *Proceedings of ML.*

[67] Scott, D., and Evans, R. 1998. Multilingual document management without translation. *Elsnews* 7.1.

[68] Shneiderman, B. 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction: Second Edition.* Addison-Wesley.

[69] Skuce, D., and Lethbridge, T. 1995. CODE4: A unified system for managing conceptual knowledge. *International Journal of Human-Computer Studies* 42(4):413–451.

[70] Tjong, E.; Sang, K.; and Buchholz, S. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL and LLL.*

[71] van Deemter, K., and Power, R. 1998. Coreference in knowledge editing. In *Proceedings of ACL-COLING Workshop on the Computational Treatment of Nominals.*

[72] van Deemter, K., and Power, R. 1999. Inclusion of picture sequences in generated documents. *ITRI Technical Report ITRI-99-01.*

[73] van Deemter, K. 1992. Towards a generalization of anaphora. *Journal of Semantics* 9:27–51.

[74] Walker, M. 1996. Limited attention and discourse structure. *Computational Linguistics* 22(2):255–264.

[75] Walker, M. 2000. Toward a model of the interaction of centering with global discourse structure. *Verbum* 22.

[76] Weiss, S., and Kulikowski, C. 1991. *Computer System That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann Publishers, Inc.

[77] Yarowsky, D. 1994. A comparison of corpus-based techniques for restoring accents in Spanish and French text. In *Proceedings of ACL Workshop on Very Large Corpora.*

Appendix A

User Study Protocols: Referring to New Entities

INSTRUCTIONS TO SUBJECTS

AUTO is a system that is used to write car repair procedures.

You are going to use 4 systems: **AUTO-one**, **AUTO-two**, **AUTO-three**, and **AUTO-four** to write sections of car repair procedures.

- Before using each system, you will be given a tutorial on how each system works.
- The tutorial will involve writing sections of a pasta recipe. The systems you use for the experiment will involve writing sections of car repair procedures.

After you complete each tutorial, Jill will start each system for you and give you 2 procedures to write.

- You can write the procedures in any order.
- You should write the procedures as fast as you can while maintaining accuracy.
- You will have a time limit of 30 minutes to write both of the procedures.

Your goal in this experiment is to write procedures that match the procedures you are given as closely as possible.

To write each car repair procedure, select **Open** from the *File* menu. Select the file that matches the part name highlighted in pink on the car repair procedure you wish to write.

When you have finished writing a procedure:

- Choose **Save** from the *File* menu. You should save the file using the part name highlighted in pink.
- Next, choose **Open** from the *File* menu, and find the file that matches the name of the part whose repair procedure you wish to write next.
- Write the second repair procedure.
- When you have finished writing ***both*** of the procedures, choose **Quit** from the *File* menu.
- Let Jill know that you are done, and she will have you fill out a short questionnaire before moving on to the tutorial for the next system.

EXAMPLE TUTORIAL: cat+vo

Writing a Pasta Recipe

This tutorial will teach you how to use a system like the one you will use in the experiment. The experiment involves writing car repair procedures; however, you will learn to use the experimental systems by writing a pasta recipe.

WRITING THE PASTA RECIPE

Figure A.1 shows the pasta recipe you will write:

Figure A.1: Pasta Recipe Description.

Part of the recipe has already been written for you. You just need to fill in the missing parts. Here is how to write the recipe:

1. First, open the template that you are provided with. It's called *pasta* since this is the term highlighted in pink at the top of the recipe. Open this file by selecting **Open** from the *File* menu and then clicking on *pasta*. You will see that part of the recipe has already been written for you. You need to fill in the parts that are missing.

   The screen should look like Figure A.2:

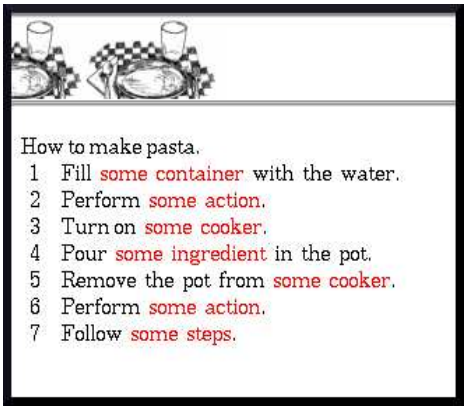


Figure A.2: Initial Pasta Recipe Template.

2. With the mouse, click once on the red words some container. The words become pink indicating that you have clicked on them with the mouse.

3. To the left, you will see a menu appear. Use the mouse to click once on the option **Common Food Holders** since a pot holds food, and we will assume that pots are commonly used when cooking.

4. Now, a menu of containers will appear. Use the mouse to click once on **pot** since the recipe says that we need to first put water in the pot. You will notice that the recipe has been updated based on your selection. The recipe should now look like Figure A.3:

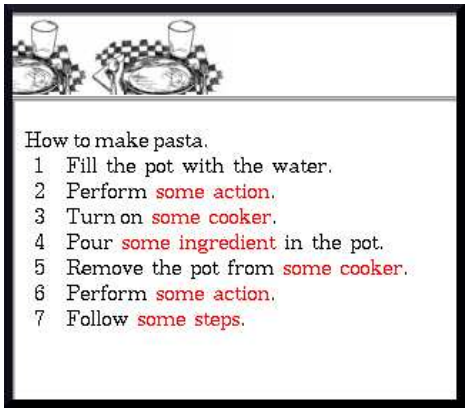


Figure A.3: Pasta Recipe Updated with **Pot** Selection.

5. Now, specify the second step by clicking on the red words some action with the mouse.

6. In the menu that appears to the left, click on the option **Common Actions** since we will again assume that *place* is an action that occurs often in recipes. Also, click on **All Actions**, and you will notice that *place* appears in this menu as well.

7. From either the **Common Actions** menu or the **All Actions** menu, click once on **place** since the next step is to put the pot on the stove. The recipe should now look like A.4:

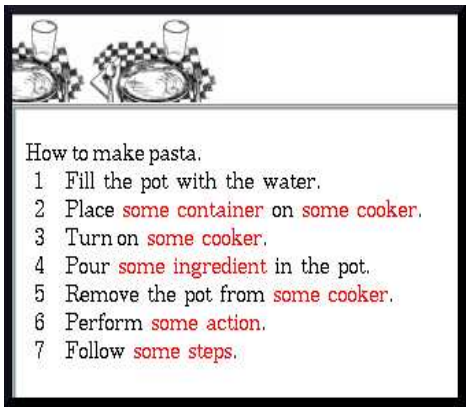


Figure A.4: Pasta Recipe Updated with **Place** Selection.

8. Now, specify the rest of the recipe based on the description that you have been provided with.

9. Once you have completed the recipe, show it to Jill.

10. Choose **save** from the *File* menu. For the name of the recipe, type *pasta* since this is the term highlighted in pink at the top of the recipe.

IF YOU MAKE A MISTAKE

Mistakes are easy to correct! If you realize your mistake right after making it, you can choose **Undo** from the *Control* menu. This simply undoes the last menu selection you chose. Try it once to see how it works. Note that you can only undo your previous operation. If you continue to choose **Undo** from the *Control* menu, nothing will happen.

If you want to correct a mistake that you made a while ago, click on the words that you want to correct. For example, let's say that you want to change the first step to "Pour water in the pot." Click on the word **fill** with the mouse. You will see a menu to the left with the option **General Action**. Click on this option, and a menu with the option **some action** will appear. Click on this option, and the recipe will look like it did before you made the mistake. Now, you can specify a **pour** action.

KEEP IN MIND

- You can choose objects to include in the recipe in ***any order***.
- You do not have to specify a value for all of the red words. For example, in this recipe, we only wanted to specify the ingredient to drain (the pasta) for Step 6, not what to drain it with.
- Don't worry if you need to choose the same object again. For example, assume that the system knows you are talking about the same *pot* in Steps 1, 2, 4, and 5.
- Let Jill know if you have any questions!

Your completed pasta recipe should be similar to the one in Figure A.5:

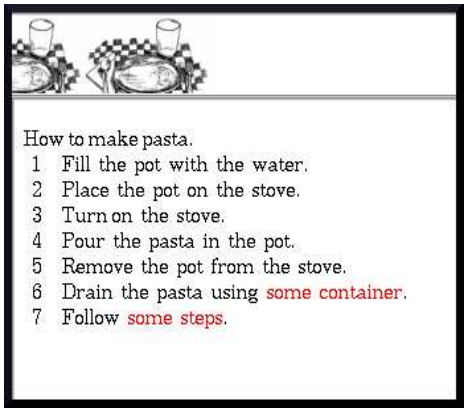


Figure A.5: Sample Completed Pasta Recipe.

EXPERIMENT QUESTIONNAIRE

Subject ID:_____ Questionnaire Number:_____

1. Rate this system on a scale of 1–5 based on how hard it was to use. (1 = very easy; 5 = very hard)

2. What made the system easy or hard to use? Write a "+" after something that made it easy to use and "–" after something that made it hard to use.

3. Rate this system on a scale of 1–5 based on how quickly you were able to write repair procedures using it. (1 = very quickly; 5 = very slowly)

4. What about the system allowed you to write repair procedures quickly or slowly? Write a "+" after something that allowed you to write them quickly and "–" after something that allowed you to write them slowly.

5. Were you ever surprised by something the system did? If so, what surprised you, and why did it surprise you?

6. If you have any comments about **any** of the systems that you used, write them in the space below.[1]

[1] This question only appeared on the questionnaire about the last system that the subject used.

Appendix B

User Study Protocols: Referring to Already Mentioned Entities

---

INSTRUCTIONS TO SUBJECTS

AUTO is a system that is used to write car repair procedures.

You are going to use 3 systems: **AUTO-one**, **AUTO-two**, and **AUTO-three** to write sections of car repair procedures.

- Before using each system, you will be given a tutorial on how each system works.
- The tutorial will involve writing sections of a procedure for assembling and disassembling a bike. The systems you use for the experiment will involve writing sections of car repair procedures.

After you complete each tutorial, Jill will start each system for you and give you a procedure to write.

- You should write the procedure as fast as you can while maintaining accuracy.
- You will have a time limit of 30 minutes to write the procedure.

Your goal in this experiment is to write a procedure that matches the procedure you are given as closely as possible.

To write a car repair procedure, select **Open** from the *File* menu. Select the file that matches the part name highlighted in green on the car repair procedure you wish to write.

When you have finished writing a procedure:

- Choose **Save** from the *File* menu. You should save the file using the part name highlighted in green.
- Choose **Quit** from the *File* menu.
- Let Jill know that you are done, and she will have you fill out a short questionnaire before moving on to the tutorial for the next system.

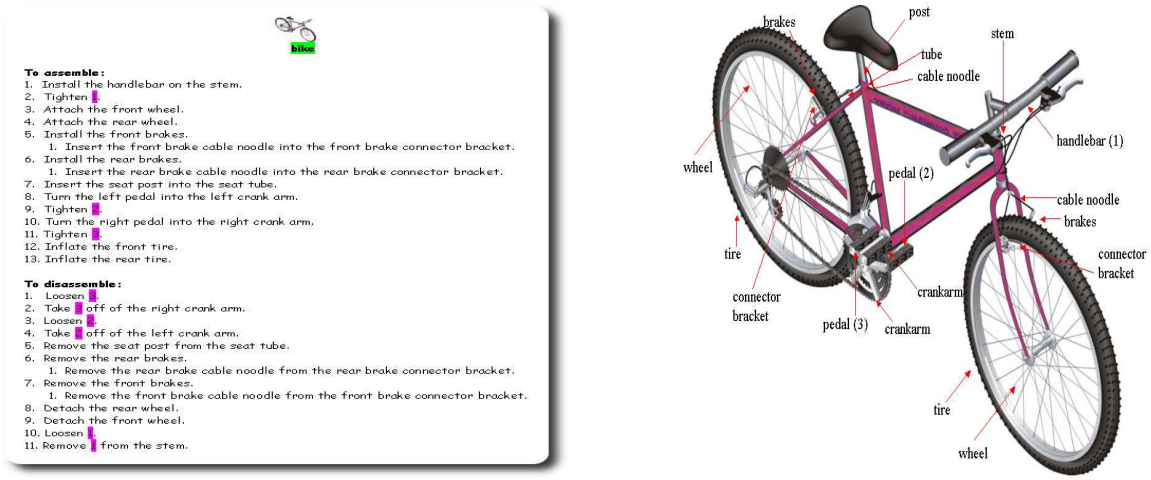
EXAMPLE TUTORIAL: disc

Writing a Bike Procedure

This tutorial will teach you how to use a system like the one you will use in the experiment. The experiment involves writing car repair procedures; however, you will learn to use the experimental systems by writing a procedure for assembling and disassembling a bike.

WRITING THE BIKE PROCEDURE

You have been given two documents: the partly specified task description shown in Figure B.1(a) and the diagram shown in Figure B.1(b). You will use the diagram to specify the unspecified parts, indicated by numbers highlighted in pink in the task description. These numbers correspond to the parts followed by numbers in the diagram.



(a) (b)

Figure B.1: Description (a) and Diagram (b) for the Bike Procedure.

Here is how to write the procedure:

1. First, open the template that you are provided with. It's called *bike* since this is the term highlighted in green at the top of the procedure. Open this file by selecting **Open** from the *File* menu and then clicking on *bike*. You will see that part of the procedure has already been written for you. You need to fill in the parts indicated by pink highlighting in the description in Figure B.1(a).

   The screen should look like Figure B.2:

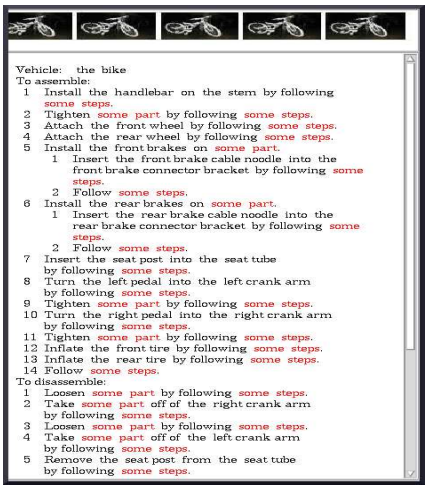


Figure B.2: Portion of Initial Bike Procedure Template.

2. The description indicates that the first part that you need to specify is in Step 2 of the bike assembly procedure. This part is indicated by the number 1. Find the part labeled with the number 1 in the bike diagram. The diagram indicates that this part is the **handlebar**.

3. With the mouse, click once on the red words some part in Step 2 of the assembly procedure, the location where the **handlebar** must be specified. The words become pink indicating that you have clicked on them with the mouse.

4. To the left, you will see a menu appear. The **Already Mentioned Parts** menu is pink indicating that its contents are displayed in the middle menu. **When you are using this version of the system, you will be making all of your selections from the Already Mentioned Parts menu.**

5. Find a selection in the middle menu that corresponds to the **handlebar**. Use the mouse to click once on the word **handlebar**. You will notice that the bike

procedure has been updated based on your selection. The procedure should now look like Figure B.3:

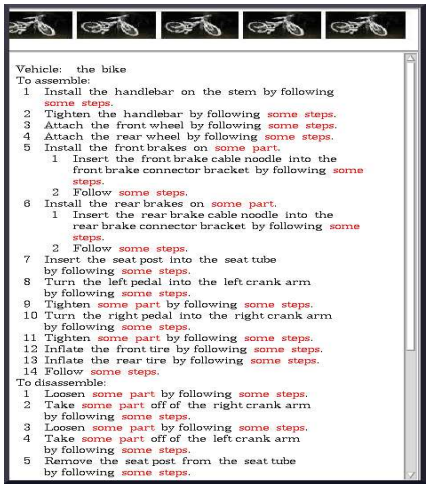


Figure B.3: Bike Procedure Updated with Repeated Mention of **Handlebar**.

6. The next unspecified part, number 2, is in Step 9. Find the part number 2 in the diagram. The diagram indicates that this part is the **pedal**. Click on the words some part in Step 9 with the mouse. The **Already Mentioned Parts** menu is pink indicating that its contents are displayed in the middle menu. There is one pedal in the menu–the **left pedal**. The diagram indicates that this is the correct pedal.

7. So, from the menu, click once on **left pedal**. The procedure should now look like B.4:

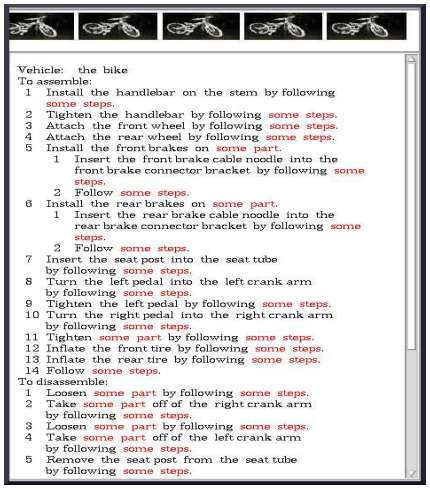


Figure B.4: Bike Procedure Updated with Repeated Mention of **Left Pedal**.

8. The next unspecified part, number 3, is in Step 10. Find the part number 3 in the diagram. The diagram indicates that this part is the **pedal**. Click on the words some part in Step 10 with the mouse. The **Already Mentioned Parts** menu is pink indicating that its contents are displayed in the middle menu. There are two pedals: the **left pedal** and the **right pedal**. The diagram indicates that you need to specify the **right pedal** in this step.

9. From the menu, click once on **right pedal**. The procedure should now look like B.5:

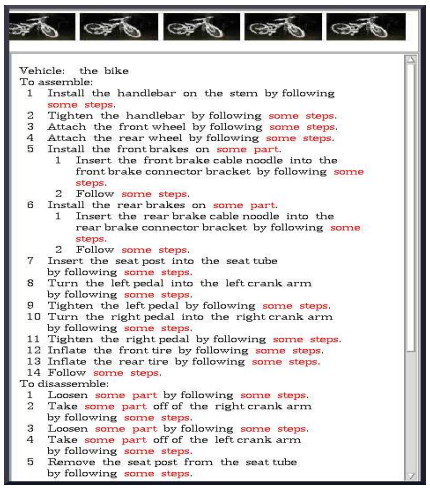


Figure B.5: Bike Procedure Updated with Repeated Mention of **Right Pedal**.

10. Now, specify the rest of the parts highlighted in pink in description using the diagram that you have been provided with.

11. Once you have completed the procedure, show it to Jill.

12. Choose **save** from the *File* menu. For the name of the procedure, type *bike* since this is the term highlighted in green at the top of the procedure.

IF YOU MAKE A MISTAKE

Mistakes are easy to correct! If you realize your mistake right after making it, you can choose **Undo** from the *Control* menu. This simply undoes the last menu selection you chose. Try it once to see how it works. Note that you can only undo your previous operation. If you continue to choose **Undo** from the *Control* menu, nothing will happen.

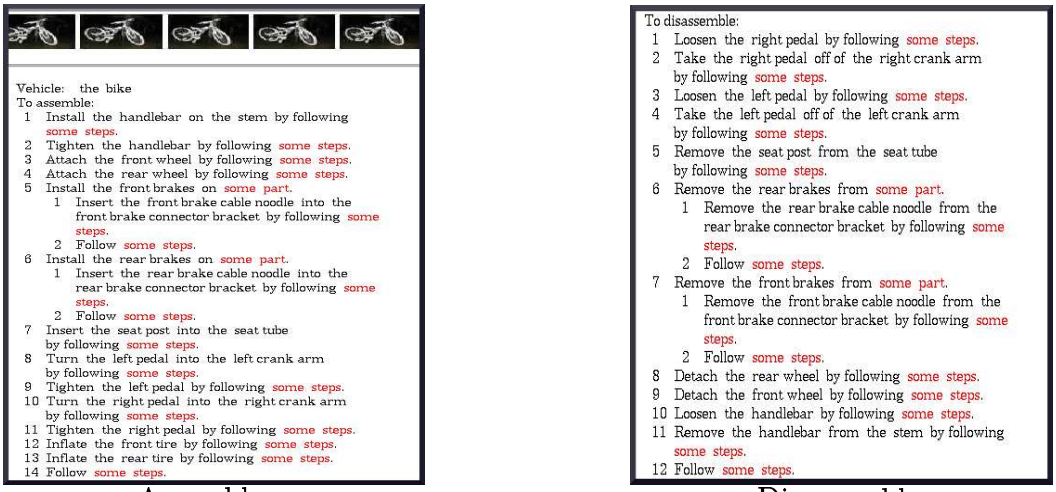
If you want to correct a mistake that you made a while ago, click on the words that you want to correct. For example, let's say that you want to change the first

step to "Install the handlebar on the frame." Click on the words **the stem** with the mouse. You will see a menu to the left with the option **General Part** and the option **some part**. Click on this option, and the recipe will look like it did before you made the mistake. Now, you can specify a **frame** part.

KEEP IN MIND

- You can choose objects to include in the procedure in ***any order***.
- For this version of the system, you should only make selections from the **Already Mentioned Parts** menu.
- You do not have to specify a value for all of the red words. For example, in this procedure, we only wanted to specify the part to install (the front brakes) for Step 5, not what to install them on.
- The same numbers indicate the same parts. For example, the number 1 in Steps 10 and 11 of the disassembly procedure indicates that the same **handlebar** is being referred to.
- Let Jill know if you have any questions!

Your completed bike procedure should be similar to the one in Figure B.6:



Vehicle: the bike
To assemble:
1 Install the handlebar on the stem by following some steps.
2 Tighten the handlebar by following some steps.
3 Attach the front wheel by following some steps.
4 Attach the rear wheel by following some steps.
5 Install the front brakes on some part.
  1 Insert the front brake cable noodle into the front brake connector bracket by following some steps.
  2 Follow some steps.
6 Install the rear brakes on some part.
  1 Insert the rear brake cable noodle into the rear brake connector bracket by following some steps.
  2 Follow some steps.
7 Insert the seat post into the seat tube by following some steps.
8 Turn the left pedal into the left crank arm by following some steps.
9 Tighten the left pedal by following some steps.
10 Turn the right pedal into the right crank arm by following some steps.
11 Tighten the right pedal by following some steps.
12 Inflate the front tire by following some steps.
13 Inflate the rear tire by following some steps.
14 Follow some steps.

Assembly

To disassemble:
1 Loosen the right pedal by following some steps.
2 Take the right pedal off of the right crank arm by following some steps.
3 Loosen the left pedal by following some steps.
4 Take the left pedal off of the left crank arm by following some steps.
5 Remove the seat post from the seat tube by following some steps.
6 Remove the rear brakes from some part.
  1 Remove the rear brake cable noodle from the rear brake connector bracket by following some steps.
  2 Follow some steps.
7 Remove the front brakes from some part.
  1 Remove the front brake cable noodle from the front brake connector bracket by following some steps.
  2 Follow some steps.
8 Detach the rear wheel by following some steps.
9 Detach the front wheel by following some steps.
10 Loosen the handlebar by following some steps.
11 Remove the handlebar from the stem by following some steps.
12 Follow some steps.

Disassembly

Figure B.6: Sample Completed Bike Procedure.

EXPERIMENT QUESTIONNAIRE

Subject ID:_____ Questionnaire Number:_____

1. Rate this system on a scale of 1–5 based on how hard it was to use. (1 = very easy; 5 = very hard)

2. What made the system easy or hard to use? Write a "+" after something that made it easy to use and "–" after something that made it hard to use.

3. Rate this system on a scale of 1–5 based on how quickly you were able to write repair procedures using it. (1 = very quickly; 5 = very slowly)

4. What about the system allowed you to write repair procedures quickly or slowly? Write a "+" after something that allowed you to write them quickly and "–" after something that allowed you to write them slowly.

5. Were you ever surprised by something the system did? If so, what surprised you, and why did it surprise you?

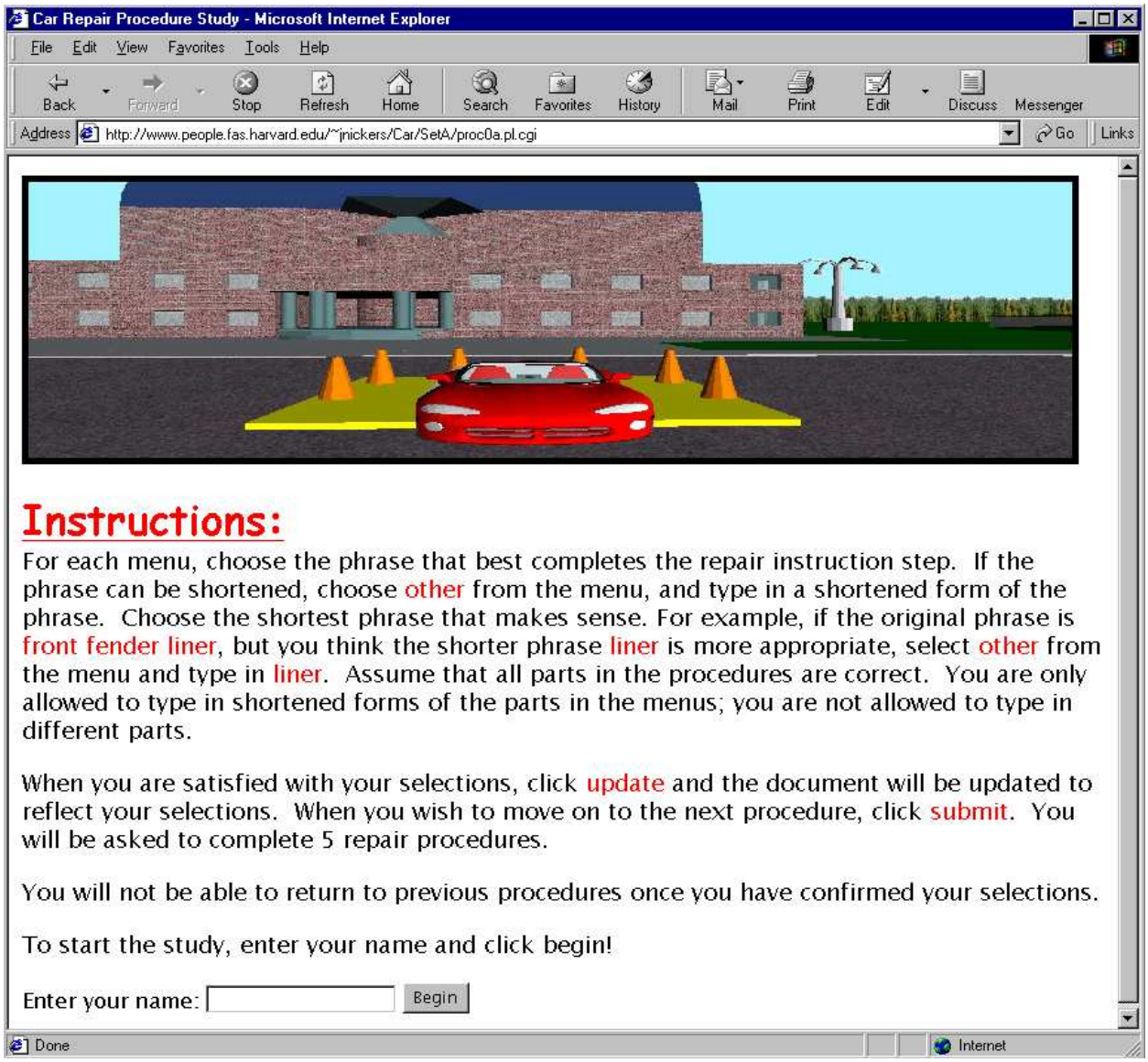
6. If you have any comments about **any** of the systems that you used, write them in the space below.[1]

[1] This question only appeared on the questionnaire about the last system that the subject used.

Appendix C

Data Collection Exercise Protocols: Obtaining Gold-Standard Referring Expressions

INSTRUCTIONAL WEBSITE



EXAMPLE PROCEDURE